Peter Norton's COPY Secrets...page 26

DOS RESOURCE GUIDE

The PC Productivity Magazine

AN IDG COMMUNICATIONS PUBLICATION
NUMBER 6
NOVEMBER 1992
$4.95
CANADA $5.95

13 Ways To Boost Your DOS I.Q.

Tips On:

- SHELL
- PRINT
- TREE
- DOSKEY

and More!

Write Your Own Batch Files: Part 1

All About

FILE & DISK COMPRESSION

DRDOS: The DOS Alternative

Painless Communications

- Modems
- BBS's
- On-line Services

RUNNING DOS WITH WINDOWS

BY BRIAN LIVINGSTON

AN IDG COMMUNICATIONS PUBLICATION

Learn About Computers the Fun and Easy Way with IDG's ...For Dummies Series

#1 Computer Books For Beginners!

Packed with useful information, yet written in everyday language, *...For Dummies* Computer Books™ explain the hows and whys of computers in a lighthearted and entertaining manner. Guaranteed to appeal to even the most reluctant computer user!

Only $16.95 Each!

DOS For Dummies

by Dan Gookin

Blockbuster Bestseller! Explains DOS essentials with humor and style. Covers DOS 2 - 5.

ISBN: 1-878058-25-8
$16.95

PCs For Dummies

by Dan Gookin and Andy Rathbone

At last, a guide to PC configuration, upgrading, and repair that everyone can understand.

ISBN: 1-878058-51-7
$16.95

"If all this talk of memory and megabytes has given you headache, take two aspirin and, in the morning, buy a copy of DOS for Dummies." —L. R. Shannon, The New York Times
DOS FOR DUMMIES
BY DAN GOOKIN
Your First Aid Kit for Dealing with DOS
Look Up the Problem—Find the Answer
What to Do When Bad Things Happen
A Reference for the Rest of Us!
Covers DOS 5
"This is a really fun way to learn WordPerfect"
WordPerfect for DUMMIES
Covers WordPerfect 4.2 Through 5.1!
by Dan Gookin
A Reference for the Rest of Us!
A Hunt & Peck Guide to WordPerfect
Look Up Your Problem—Find the Answer
Brush Up on the Basics and Learn Advanced Tricks Too
An Excellent Companion to the #1 Bestseller DOS for Dummies!
NEW!
PCs FOR DUMMIES
by Dan Gookin & Andy Rathbone
A Reference for the Rest of Us!
The Fun Way to Learn & Master Your PC
Your First Aid Kit to PC Set-Up, Maintenance, & Troubleshooting
What to Do When Bad Things Happen
Covers the Latest PC Hardware & Software
"Pogue is a wonderful teacher. If he can communicate with someone as anti-technical as myself, he can teach anybody." —Guy Takeo, Bestselling author of House Thy Father
NEW!
MACS FOR DUMMIES
by David Pogue
The Fun Way to Learn & Master Your Macintosh
How to Plug in Your Mac (And What to Do Next)
Your Instant Cure for Chronic Computer Phobia
A Reference for the Rest of Us!
IDG BOOKS

WordPerfect For Dummies

by Dan Gookin

Here are all the basics you need (and then some) for mastering WordPerfect quickly.

ISBN: 1-878058-52-5
$16.95

Macs For Dummies

by David Pogue, *Macworld* Magazine Columnist

Finally, a friendly guide to plugging in and learning how to use your Mac!

ISBN: 1-878058-53-3
$16.95

Get Smart! Order Now!

Call 1-800-762-2974

7 am to 5 pm PST or complete and mail this form.

Yes! Please send me:	Qty.	Price	Tax*	Ship	Total
______ *For Dummies*	______	$16.95	______	$4.00	______
______ *For Dummies*	______	$16.95	______	$4.00	______

UPS Shipping & Handling ($4 per book) *CA residents add applicable sales tax. IN residents add 5% tax. Canadian residents add 7% GST tax.

Name ______
Company Name ______
Address ______
City, State, Zip ______
Phone ______

Payment method (payment with credit card number or check must accompany your order:)
☐ Check enclosed ☐ Visa/Mastercard/American Express

Card # ______ Expiration Date ______
Signature ______

Inquire about rush delivery service (an extra charge will apply).
Corporate trainers quantity discounts available.

DRGDUM

IDG BOOKS
The World of Computer Knowledge

IDG Books Worldwide
155 Bovet Road, Suite 610
San Mateo, CA 94402
(415) 312-0608
(800) 762-2974

DOS RESOURCE GUIDE

EDITOR
Michael J. Comendul
MANAGING EDITOR
Marilyn McMaster
SENIOR EDITOR
David Essex
EDITOR-AT-LARGE
Eric Maloney
ARTICLE EDITORS
Patricia Cummings, Beth Jala
TECHNICAL EDITORS
Hardin Brothers
Mare-Anne Jarvela
John Wolfskill
CONTRIBUTING WRITERS
Ruth Ashley, David Bannon, Antonio & Alfonso De Pasquale, J. Fernandez, Dan Gookin, Jan Norman Johnsen, James Langdell, Brian Livingston, Patrick Marshall, Jack Nimersheim, and Peter Norton
ART DIRECTOR/ MANUFACTURING MANAGER
Lou Ann Morin
ELECTRONIC PUBLISHING
Lindy P. Greenwood

How to contact us: *DOS Resource Guide* welcomes your comments, DOS tips, and article submissions. Submit them care of our offices at the address below, or by CompuServe: 75300,2361; or MCI Mail: 469-7260.

DOS Resource Guide (ISSN: 1056-7364) is published six times per year by Business Computer Publishing Inc., an IDG company. Business and editorial offices are located at 80 Elm St., Peterborough, NH 03458. 603-924-0100. Entire contents copyright 1992, by Business Computer Publishing Inc. No part of this publication may be printed or otherwise reproduced without written permission of the publisher. Subscriptions to *DOS Resource Guide* are available in the U.S. for $27 annually. *DOS Resource Guide* makes every effort to verify its articles and program listings but assumes no responsibility for damages due to errors or omissions.

Back issues of *DOS Resource Guide* are available in the U.S. for $5.95, postage paid. Canadian and Mexican residents pay $6.95 per issue (including taxes, postage and handling). All other foreign orders pay $8 (surface mail), $13 (air mail). Call: 800-238-6580 (9 a.m. to 5 p.m. EST).

Business Computer Publishing Inc.

PRESIDENT/PUBLISHER
Jeff DeTray

NEWSSTAND CIRCULATION
Kemco Publishers Services
(603) 924-0224

IDG COMMUNICATIONS *DOS Resource Guide* is a publication of International Data Group, the world's leader in information services on information technology.

Starting With What We Know

Before the meeting, each editor draws out a few of the wrinkled pale-brown sheets from a pile strewn across the center of the table. These are reader surveys—questionnaires that tell us about the equipment you own, the ways in which you use your computers, and your thoughts about the magazine. These surveys represent only a handful of the responses we have received over the last several months.

Speculating from incomplete voter returns is dangerous—especially this year—but I'll venture forward. The first, and probably most remarkable, conclusion to draw about our readers is your undeniable diversity, not only in jobs—we've heard from geologists, doctors, students, teachers, engineers, and retirees—but in your computing skills. You work at all levels of DOS on every kind of equipment. Some readers are learning Basic at home on outdated double-floppy computers, others manage complex networks using 33MHz 386DXs. (In fact, many of you have better equipment than I.)

This diversity of experience presents a thorny editorial problem. How are all of you finding happiness in the same magazine? Can it continue?

I think so. By staying our editorial course of speaking plainly about what are traditionally hazy technical topics, *DOS Resource Guide* can reach a broad spectrum of amateurs and computing professionals. You've told us that much. With consistency, readers have said what most attracts them to the magazine is the "clarity" of our articles, their "lack of jargon," the "detailed explanations." One reader writes, "I can understand the language"; says another: "It doesn't make me feel dumb."

My conclusion: Many people, even those who have been dealing with computers for years, are as fuzzy as any novice when it comes to deciphering the much misused vocabulary of computing: shadow RAM and cache RAM, real mode and protected mode, multitasking and task swapping. Further, among the computer magazines in today's market, many do a thorough job of presenting what's new in computing, but few unravel the mysteries of *how* computers work.

That's where *DOS Resource Guide* can help. We assign our writers to find a limitation in DOS and work around it, or to find a computer problem and solve it. In future issues, as our cover story conveys, we will be extending DOS's reach to communications programs, memory managers, task swappers, and general utilities.

Your response to our issues is gratifying, passionate, even humorous. One retired reader, whose survey is black with comments, wants a questionnaire with more blank space next time. Another reader, who may have us confused with Oprah Winfrey, likes the "frankness" of our articles. One other opinion is clear: You'd like to see more of *DOS Resource Guide*.

We've heard you: With this issue, *DOS Resource Guide* will be published every other month.

We'll see you then.

Michael J. Comendul

DOS RESOURCE GUIDE

DOS

Give DOS 5.0 a Friendlier Face 17

Paint your DOS screen with a splash of bold color and, in the process, learn what you can and can't do to personalize DOS 5.0's shell.

by Jack Nimersheim

Debug Does Disks 21

Part II of this series moves beyond recreational exercises, revealing the secrets of reading from and writing to disks.

by Dan Gookin

Advanced Class in Copying Files 26

If you've got a problem with COPY, who better to consult than Peter Norton, creator of some of today's best-selling DOS utilities.

by Peter Norton, Ruth Ashley, and Judi Fernandez

BAT

Programming in DOS's Batch Language 37

Once you learn how simple it is to write your own batch files, you'll wonder why you waited so long. Part I in a series.

by Hardin Brothers

Command Performance 41

If you long to tell DOS where to go and what to do—with the press of a key—this tutorial on reprogramming your keyboard's function keys can help.

by Antonio and Alfonso De Pasquale

FYI

How to Beat Modem Anxiety 45

You can overcome your fear of telecommunications. Today's software and modems make using bulletin boards and exchanging files easier than you think.

by Jack Nimersheim

DR DOS 6.0: A Most Worthy Opponent 51

A built-in disk "defragger" and automated task switching are two reasons you might consider buying the latest version of this upstart operating system.

by John Wolfskill

More Disk Space for Free 56

Shrink that file; crunch that disk. Why buy a new hard drive when you can fit more data on the one you have?

by Patrick Marshall

BAS

The Importance of Good Thinking 61

Before you leap into writing a QBasic program to solve a computing problem, use your head.

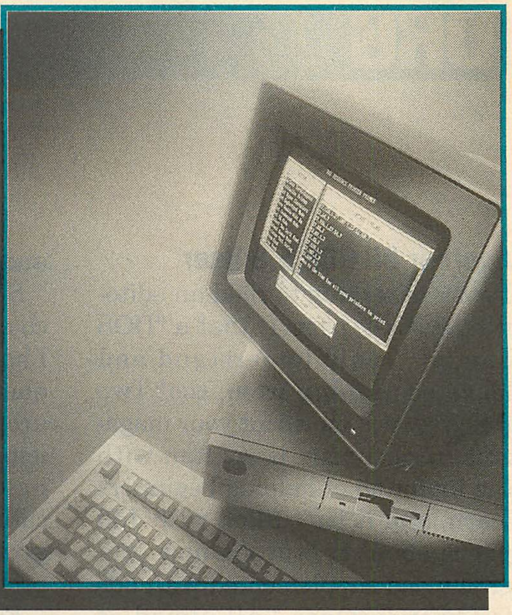
by John Wolfskill

Building a QBasic Database 65

Keep track of names and addresses with this easy-to-use (and easy-to-customize) program.

by David Bannon

Number 6
November 1992



COVER PHOTO BY LARRY DUNN

DRG

Editorial ... 1

Letters to the Editor ... 4

Tips from Readers ... 7

Best of the Batch ... 10

Q&A: DOS, Disks, and Dilemmas ... 12

TIP

For Your Eyes Only
Safeguard files by adding a secret character to your filenames. ... 24

Speedy Directory Searches
Make DOS more responsive during path searches. ... 25

Gone, But Not Forgotten
Prevent DOS 5.0's deleted files from getting in your way. ... 32

Winning the SHELL Game
How to use SHELL to load a command processor. ... 34

Stars Be Gone
List, copy, or delete files with fewer keystrokes. ... 38

Time and Date Bypass
Keep things moving even when DOS displays the time and date during start-up. ... 39

The Key to More DOSKEY
Expand the size of your DOSKEY buffer. ... 42

Background Printing in DOS
Teach PRINT to work in the background while you get on with other tasks. ... 43

Type it PDQ!
Create a text file on the double. ... 50

Look: No Hands
Redirecting input and output keeps batch files in business. ... 52

C the TREE. C the TREE grow.
Using TREE to make sense of your directory structure. ... 53

Make Your Keyboard Fly
Rev up your keyboard's response rate. ... 54

Helpful Associations
Launch an application and related file simultaneously. ... 60

WIN

Using DOS Under Windows ... 70
Discover the tricks to achieving peaceful coexistence between DOS and Windows.
by Brian Livingston

Q&A: Windows ... 77
by Jan Norman Johnsen

DRG LETTERS TO THE EDITOR

Another DR DOS 5.0 User

It was refreshing to read an editorial (*DRG*#4, page 1) that a "DOS dummy" could understand and learn something from, too! Two things struck me about your magazine. It obviously is written with the intention of helping beginners learn, and it isn't filled with 150 pages about rapturous new programs and supplemented with ads for "wonderware," all aimed at eliminating the pain of learning DOS 5.0.

Like Ivan Sinkler of Alameda, Calif. (*DRG*#4, Letters to the Editor, page 3), I have a system that operates under DR DOS 5.0. How about an article on the main differences between the MS-DOS and DR DOS versions, or the ins and outs of the DR DOS system?

Since my system was a gift, purchased by others 1,000 miles away, I have no dealer to turn to with my questions. Thinking others in the area might have the same problem, I ran a notice in the paper for new PC users, and 14 people called. Now we have our own PC group to wrestle with common problems and trade tips.

For my money, you are "right-on." Don't get any more technical than you are!

Robert N. Walsh
Vero Beach, Fla.

Windows Doesn't Wash

I recently picked up your fourth issue, and I'm amazed that someone finally has written a magazine with articles that are comprehensible to a novice computer user such as myself. I've read lots of magazines trying to better understand many different aspects of my computer, but frequently the articles aren't worth reading.

I love your magazine; it is superb. I applaud your level of explanations and instructions! Frankly, I'm sort of tired of reading all about this "great" but yet "ungreat" Windows program, and it was nice not to see your entire magazine devoted to Windows. Personally, thus far I haven't figured out any reason to purchase the program. Keep up the great articles!

Judy Geyer
Galion, Ohio

Don't Leave Tandy Users Out

You cover many separate topics and subtopics in each issue of *DOS Resource Guide*. It is really more of a manual than a magazine. Readers like me, who have underdeveloped photographic memories and can remember general ideas but not the details, would appreciate an index listing the page on which each topic appears.

Also, some readers (me included) own Tandy clones of the IBM AT. Our machines came with a primitive but usable graphical user interface called DeskMate, which runs a number of pieces of semiuseful software, such as a text editor, a spreadsheet, and a filer. You previously have published information on non-MS-DOS software, such as Windows, that is closely coupled to MS-DOS.

Readers who are interfacing with MS-DOS by means of the Tandy GUI, or who are creating and reading files with the Tandy text editor or other Tandy soft-

Let Us Hear From You

DOS Resource Guide wants to know about your computing trials and tribulations. The editors welcome your correspondence and encourage you to share your DOS tips or criticize ours.

From among the letters and batch files we receive, the editors will choose a "Best of Batch" selection that could earn you $50 and a published credit.

So whether you work as a network administrator, a high school teacher, or just hang out at the command prompt, you have something to share with *DOS Resource Guide* readers. Send your criticism, your praise, and your best batch files to:

DOS Resource Guide
80 Elm St.
Peterborough, NH 03458

or reach us through CompuServe: 75300,2361 ware, would welcome Tandy-oriented MS-DOS hints.

David Ferrier
Edmonton, Alberta

Keep the Lid on Ads

I have purchased the first three issues of your *DOS Resource Guide* and I am writing to you to give you a reader's point of view.

I thought the first and third issues were great. In my job, I go around from office to office to repair PCs. I also load software, write batch programs for users, and upgrade hardware. In order to keep up with the job, I read many magazines and books on PCs. I must say, your magazine is one of high caliber and is now an essential part of my own resource library.

You are doing wonderful things with batch files, DOS commands, and insights on memory. But I think the software advertising should be kept to a few pages.

I have shared this magazine with many of my friends and co-workers and they, too, agree it's a great resource for PCs.

I'm looking forward to your next issue. Keep up the good work.

Debra Marie Wintermyer
New Cumberland, Penn.

Whys and Wherefores

Great! Just when I thought my nights were going to be filled with endless hours of reading very boring and unintelligible manuals, I happened upon your magazine. Now, no more sessions with my nose in the manuals, eyes riveted to the screen, and getting to bed at 4:30 a.m. with a feeling of accomplishing absolutely nothing. Why is it that people who write manuals assume the reader (often a first-time PC user) understands the reason for a statement such as, "This command does 'this and that,'" and they never explain why or when the user would want to use the command?

Thank goodness for a publication like yours. You take the time to explain not only the why or when but the "how to" of it all. I hope you never assume the reader knows what you are talking about. Here's to a long and prosperous publication run. Your magazine definitely was written for me. Thanks.

Ed Roberts
Sierra Vista, Ariz.

OMISSIONS AND CORRECTIONS

Moving Concerns

I enjoyed *DRG*#4 so much that I sent for the previous issue and was not disappointed. I thoroughly enjoyed all the articles.

However, in *DRG*#3 (page 68), I found a small error in Program Listing 2, GETKEY.SCR. The seventh line should read:

```
MOV AH,4C
```

T.F. Finnegan
Poughkeepsie, N.Y.

WIN Correction

I purchased *DRG*#4 and read it with enthusiasm. The articles were interesting, well-written, and had just the right depth. But I noticed an error. In WIN.BAT (Q&A: Windows, page 76), a batch file that prevents you from accidentally running Windows 3.0 a second time or ending a computing session when applications might be running, you forgot to put the percent signs around *WIN*. The second line should read:

```
IF "%WIN%"=="Y" GOTO INWIN
```

This batch file is a great idea that I adapted to my Turbo C, Quick Basic, and QBasic batch files.

Dan Schellenger
Riverside, Calif.

Rebox It!

The table in the article "Box It Up" (*DRG*#5, page 64) contains an error in the key assignments. If you follow the article's instructions, the Del key is changed to a single or a double vertical line (not the corner brackets indicated). The columns also are mislabeled. The first column lists the key's function, the latter two list the ASCII characters assigned to the various keys by SBOX.BAT and DBOX.BAT.

Neil Burnett
Titusville, Fla.

Elementary Definition

The magazine has received a few letters from readers regarding Program Listing 1 for PHONE.BAS ("Elementary Steps to Programming," DRG#4, *page 65). A "duplicate definition" error message in line 16 or 17 seems to be the problem. This message apparently is displayed only if you use Quick Basic 4.0. The program should work fine with QBasic or Quick Basic 4.5.*

—Eds.

CONFIG.SYS Changes

The listings in reader Jerry Simpson's tip (DRG#4, *page 7) contain a few typos. Mr. Simpson's series of batch files reboots your com-*

puter, allowing you to change configuration files when you change applications. In Listings 1-4 the words NO CHANGE *should appear as* NOCHANGE *(omit the space). In Listing 4, the third line should read:*

```
IF "%FIG%"=="%FYG%" NOCHANGE
```

Finally, the last line in Listing 13 is missing a space. It should read:

```
TYPE MENU.TXT
```

A couple of readers spotted the problems and one, Richard Knox (see "CONFIG.SYS Revisited" on page 9), thought he had a much better method for changing your CONFIG.SYS on the fly.

—Eds.

Directory Assistance

Because of a typographical error, the listing RUN.BAT ("Best of the Batch," DRG#5*, page 9) was truncated and difficult to read. For that reason, we are reprinting the entire listing below (see the Program Listing).*

—Eds.

Program Listing. RUN.BAT lets you use a single command to run a program on any drive in any directory.

```
@ECHO OFF
IF "%1"=="" GOTO ERROR
IF "%1"=="+" GOTO ERROR1
CD\
IF "%2"=="" GOTO LOOP
FOR %%P IN (A: a:) DO IF "%1"=="%%P" GOTO DRIVE
FOR %%P IN (B: b:) DO IF "%1"=="%%P" GOTO DRIVE
FOR %%P IN (C: c:) DO IF "%1"=="%%P" GOTO DRIVE
FOR %%P IN (D: d:) DO IF "%1"=="%%P" GOTO DRIVE
FOR %%P IN (N: n:) DO IF "%1"=="%%P" GOTO DRIVE
GOTO LOOP
:DRIVE
        %1
        SHIFT
        CD\
:LOOP
        IF "%1"=="+" GOTO ERROR1
        IF "%2"=="" GOTO DONE
        IF "%2"=="+" GOTO DONE
        IF EXIST %1\NUL GOTO CHGDR
        GOTO ERROR3
:CHGDR
        CD %1
        SHIFT
        GOTO LOOP
:DONE
        IF EXIST %1.COM GOTO RUN
        IF EXIST %1.EXE GOTO RUN
        IF EXIST %1.BAT GOTO RUN
        IF EXIST %1\NUL GOTO CHGDR1
:CHGDR1
        CD %1
        IF EXIST %1.COM GOTO RUN
        IF EXIST %1.EXE GOTO RUN
        IF EXIST %1.BAT GOTO RUN
        SHIFT
        IF "%1"=="" GOTO ERROR2
:RUN
        IF "%2"=="+" SET CMD=%1 %3
        IF "%2"=="" SET CMD=%1
        %CMD%
        GOTO END
:ERROR
        CLS
        ECHO You must enter Drive Directory
        ECHO Program + Parameters
        GOTO ENDERROR
:ERROR1
        CLS
        ECHO You can enter the + only before
        ECHO command line parameters
        GOTO ENDERROR
:ERROR2
        CLS
        ECHO The executable file you have
        ECHO entered does not exist
        GOTO ENDERROR
:ERROR3
        CLS
        ECHO You have selected an invalid
        ECHO directory
:ENDERROR
        ECHO.
        PAUSE
:END
        SET CMD=
        C:
        CD\
        CLS
```

End

Hidden Files

I was just given a copy of your magazine for review. I like it. Plain language on all kinds of material is exactly what I needed. I do have a comment on Q&A: Files and Directories (*DRG*#4, page 13), though. You don't have to use DOSSHELL or the ATTRIB command to see hidden files. With DOS 5.0, you simply can issue the DIR command with a /A parameter. This will show you *all* files and directories, regardless of which attributes are set. If you want to see only the hidden files, use:

```
DIR /A:H
```

If you want to list all files, except the hidden files, type:

```
DIR /A:-H
```

This method works with any valid attribute: archive (A), directory (D), hidden (H), read-only (R), and system (S). You also can use this technique with DIR's wide (/W) and page (/P) parameters, and you can issue this command on more than one attribute at a time. The following command, for example, provides a wide directory listing of all your hidden directories:

```
DIR /W /A:HD
```

Dennis Durham
New Albany, Ind.

DIR Made Simple

I just bought my first issue of *DOS Resource Guide* and now plan to buy the three previous issues. I was waiting at the check-out stand in the grocery store and threw the magazine in my basket as an afterthought. That night, I stayed up late reading it, slept through my alarm, and was late for work. I was too tired to be really productive, so I read it some more. Anyway, I decided to share two simple batch files that I have put on our PCs at work [see Program Listings 1 and 2]. They are helpful because you can install them on your primary menu (if you use menus on your PC) and use them to list the files on a floppy disk without exiting to DOS and typing the DIR command with its /P and /W switches. To invoke Program Listing 1, you simply press the D key. Pressing D1/ invokes the second listing. Thanks for a great magazine.

Program Listing 1. D.BAT provides a list of the files in the root directory on drive A, displaying the list in wide format and pausing after each page.

```
ECHO OFF
DIR /W /P A:
PAUSE
CD\
MENU
```

End

Program Listing 2. D1.BAT provides a list of files in the root directory on drive A, pausing after each page.

```
ECHO OFF
DIR /P A:
PAUSE
CD\
MENU
```

End

Martin Schaefbauer
Yuma, Ariz.

Small But Big

First, congratulations on a very informative and useful magazine. I'm in the process of teaching some new employees the rudiments of using personal computers, and your magazine has been most helpful. It emphasizes the use of batch files. As your articles suggest, these can make life a lot easier for the DOS user. However, there is another aspect of batch files that should be considered: disk space.

DOS allocates space for file storage by clusters. On a smaller hard drive or on a floppy disk, a batch file may require one 512-byte cluster of disk space, even when the file itself occupies only 100 or fewer bytes. Larger disk drives use larger clusters because of the limited number of entries permitted by the file allocation table (FAT). In some cases, these clusters can be as large as 4K or 8K. That means the small batch file now takes up quite a lot of disk space. For instance, if the cluster size is 4K, a 100-byte batch file will waste 3,396 bytes! If you have numerous small batch files, your disk space will shrink dramatically.

One way to reduce the amount of lost storage space is to use one batch file with a variable (%N) and the GOTO command. Or you can use ERRORLEVEL to combine a collection of smaller batch files.

Murray Glazier
Arlington, Texas

Mr. Glazier is absolutely right. Because of the way DOS allocates disk space, small files occupy more than their share of hard disk real estate. You can determine the size of the clusters on a disk (DOS calls them allocation units) by running CHKDSK.

On one of our machines, the relevant line reads "2048 bytes in each *allocation unit." That's the minimum amount of disk space a file can occupy on this drive. You can prove it to yourself with a simple experiment. First, invoke DIR at the DOS prompt to display a directory. At the end of the directory display, DOS shows the number of free bytes on the drive. Write down that number. Now, copy or create a small file on the drive. One way to do this is by typing the following commands at the DOS prompt, pressing Enter after each line:*

```
COPY CON TEST.TXT
X
^Z
```

To type the third line, press Ctrl-Z and then Enter. This creates a 3-byte file called TEST.TXT. (That's 1 byte for the letter X, 1 byte for the carriage return after the X, and 1 byte for the Ctl-Z character, which is the DOS end-of-file marker.)

Now display the directory again. You'll see TEST.TXT listed with a file size of 3 bytes. But look closely at the number of bytes free. It will be smaller than its previous value by the size of one allocation unit (or cluster). On our test machine, the number of free bytes will be reduced by 2,048, even though TEST.TXT is a mere 3 bytes in length.

*Mr. Glazier's suggestion for combining several small batch files into one is a useful technique for saving disk space. We explained how to do this in "Melting Pot" (*DRG*#1, page 16) and "Take Control With Batch Files" (*DRG*#4, page 55).*

The amount of disk space saved by this method depends on the size of the allocation units on your drive (a function of the drive's capacity) and on the number and size of your batch files.

—Eds.

Move Over MOVE.BAT

I noted that the batch file MOVE.BAT in "Take Control With Batch Files" (*DRG*#4, page 59) doesn't check to see if the source file already exists on the destination disk, which means you could unwittingly overwrite a different file with the same name. To correct this gap, you need to add an IF EXISTS test to your file, which I've done in my modified version of MOVE.BAT [see Program Listing 3].

Here's how my version of MOVE.BAT works. When you type a command such as the following at the DOS prompt:

```
MOVE TEST.TXT TARGETDRV
```

the target pathname, TARGETDRV, is stored as %2, and the filename TEST.TXT is stored as %1. The third line of the program checks to see that a target pathname is specified. The fourth checks that a filename is specified; if it isn't, MOVE.BAT prompts the user to follow proper syntax. The following line then checks to make sure a file with the same name doesn't exist in TARGETDRV; if it does, the program ends. The sixth line uses XCOPY to copy the file to TARGETDRV, and then the target directory is checked again to make sure the file was moved successfully. If the move is successful, the file in the source directory is deleted. The rest of the listing contains error messages that inform the user when proper syntax is not followed.

Dan Schellenger
Riverside, Calif.

Program Listing 3. Move.BAT copies a file to a new location, then deletes the copy of the file from its old location.

```
@ECHO OFF
IF "%2"=="" GOTO ERR1
IF "%1"=="" GOTO ERR1
IF NOT EXIST %1 GOTO ERR2
IF EXIST %2\%1 GOTO ERR3
XCOPY %1 %2
IF NOT EXIST %2\%1 GOTO ERR4
DEL %1
GOTO END
:ERR1
ECHO To move a file or files to another directory,
ECHO use the syntax: MOVE filename pathname.
ECHO
GOTO END
:ERR2
ECHO %1 doesn't exist in this directory.
GOTO END
:ERR3
ECHO File already exists on target directory; move aborted.
GOTO END
:ERR4
ECHO Move failed; file(s) NOT deleted.
:END
```

End

CONFIG.SYS Revisited

Jerry Simpson's system for changing your computer's configuration settings (Tips From Readers, *DRG*#4, page 6) can be simplified by first deleting all occurrences of *CD* and *CD\\CONFIG* in Program Listings 1–14. Then, insert the following two lines after the ECHO statement in Listing 14:

```
C:
CD\
```

Next, place your TXT and BAT files in a directory mentioned in your AUTOEXEC.BAT file's PATH statement, and substitute the following command for the final line in Listing 13:

```
TYPE C:\BATCH\MENU.TXT
```

Also, you may change the IF EXIST statements to COPY statements.

I tried an alternative approach to Simpson's, using a single batch file instead of 14 short ones [see Program Listing 4]. My program, C2.BAT, uses half as many lines of code, warns you before rebooting, gives you the opportunity to reselect the current configuration, and is totally self-contained. A modified DOS prompt shows the active configuration.

Richard Knox
Reno, Nev.

Program Listing 4. C2.BAT lets you automatically reconfigure your computer's CONFIG.SYS and AUTOEXEC.BAT files.

```
@ECHO OFF
REM C2.BAT allows automated reconfiguration of your
REM computer. It requires that AUTOEXEC and CONFIG.
REM files with matching extensions reside in the
REM root directory and that C2.BAT resides in a
REM directory listed in the PATH statements of all
REM variant AUTOEXEC.BAT files.
CLS
C:
CD\
IF %1!==! GOTO NOPARAM
FOR %%F IN (n N s S) DO IF %1==%%F GOTO %1

:NOPARAM
   ECHO C2.BAT requires a valid parameter:
   ECHO.
   ECHO       Use    C2 N   (for NORMAL configuration)
   ECHO       Or     C2 S   (for SPECIAL configuration)
   GOTO END

:n
:N
   SET FYG=NOR
   GOTO TEST

:s
:S
   SET FYG=SPE

:TEST
   IF %FIG%==%FYG% GOTO END
   IF NOT EXIST CONFIG.%FYG%   GOTO MISSING
   IF NOT EXIST AUTOEXEC.%FYG% GOTO MISSING
   ECHO.
   ECHO WARNING!  Press Ctrl-C to AVOID REBOOTING to %FYG% or
   PAUSE
   COPY CONFIG.%FYG%   CONFIG.SYS    >NUL
   COPY AUTOEXEC.%FYG% AUTOEXEC.BAT  >NUL
   REBOOT

:MISSING
   ECHO.
   ECHO Error...Can't find CONFIG.%FYG% or AUTOEXEC.%FYG% in the
   ECHO          root directory. Exiting now to fix the problem.

:END
   ECHO.
```

End

Better View

The tip "More Is Less" (*DRG*#5, page 43) provides a short batch file, VIEW.BAT, that makes it easy to view ASCII files. But unfortunately, if the last page of the file being viewed is not a full screen, the batch file will end and the screen will be cleared before you have a chance to read it. Just remove the last CLS statement from the listing, and the last page will not disappear.

Daniel Schellenger
Riverside, Calif.

Safe Format

I've developed a worry-free method for dealing with FORMAT that has kept me out of trouble on more than one occasion. It prevents me from reformatting my hard disk and lets me format floppy disks in drives A and B—even if I can't remember the appropriate FORMAT switches. My technique involves two programs. The first, a version of GETKEY.COM, captures keyboard input and feeds it to the second program, a batch file called FORMAT.BAT that assumes responsibility for formatting chores (see Program Listings 1 and 2).

To use my technique on your system, start by renaming DOS's formatting utility, FORMAT.COM, as FORMAT!.COM so that when you type `FORMAT` at the DOS prompt, DOS will find and run FORMAT.BAT. Next, adjust FORMAT.BAT to suit your computer's configuration. As written, the program uses DOS 5.0 formatting commands and is set up for a system that has a 5¼-inch drive as drive A, a 3½-inch drive as drive B, and a hard drive as drive C. If drive A on your system is a 3½-inch drive and your drive B is a 5¼-inch drive, substitute *B*: for *A*: (and vice versa) in the lines that begin with a FORMAT! command. You can, of course, also change the syntax of the FORMAT! commands to comply with that of another DOS version.

When you invoke FORMAT.BAT, the program starts by checking the command line arguments to see if you specified a drive letter when invoking FORMAT. If you didn't, the program jumps to :HELP, displays the help menu, and invokes GETKEY.COM, which waits for you to press a key. When you do, GETKEY.COM returns the ASCII value of the key to the program as an Errorlevel value. The FOR loop takes each value within parentheses in turn, comparing it to the Errorlevel value returned by GETKEY.COM. When a match is found, the program sets the value of Opt to the Errorlevel value. In the following line, that value is appended to the CASE statement.

Program Listing 1. GETKEY.SCR is a Debug script file you can use to create the executable program GETKEY.COM. Type the program listing into Edlin, Edit, or a word processor that saves files in text (ASCII) format. Don't type in semicolons or remarks. Save the file as as GETKEY.SCR. Then, at the DOS prompt, type `DEBUG<GETKEY.SCR`.

```
N GETKEY.COM
A 100
MOV AH,0     ;ROM BIOS service
INT 16       ;Read next keystroke into AL
AND AL,DF    ;Force to uppercase
MOV AH,4C    ;DOS function
INT 21       ;Terminate with exit code in AL

RCX
10
W
Q
```

End

I use a version of GETKEY.COM that accepts upper- or lowercase input and outputs an error level that matches the value of the uppercase letter. (Eliminating a check for lowercase input keeps the FOR command line relatively short.) Because the value of Errorlevel is valid if the value of the key you pressed is equal to or greater than the value against which it is compared, I use zero to collect keystrokes with ASCII values lower than 64 (A); I use 74 (J) to collect any keystroke greater than I, the help menu's Exit option. If you press a key outside the A–I range, the program echoes an error message to the screen and pauses so that you can see what happened and try again.

If you specify drive letter A or B when invoking FORMAT.BAT, the program goes to the Forma routine, which accepts as many as four parameters. Including this routine permits experienced users to bypass the help menu and specify formatting switches—from the usual, such as /F, to the more esoteric, such /U (unconditional format).

Both experienced and inexperienced users are protected from reformatting their hard disks by mistake. If you type:

```
FORMAT C:
```

at the DOS prompt, FORMAT.BAT displays an error message and ends.

GETKEY.COM is a useful little program. You can use it in any situation in which a menu or help screen is applicable, you have fewer than 26 selections, and you are willing to use letters to designate those selections.

Dan Schellenger
Riverside, Calif.

Program Listing 2. FORMAT.BAT takes the worry out of using the FORMAT command.

```
@ECHO OFF
IF "%1"=="" GOTO HELP
IF "%1"=="A:" GOTO FORMA
IF "%1"=="a:" GOTO FORMA
IF "%1"=="B:" GOTO FORMA
IF "%1"=="b:" GOTO FORMA
IF "%1"=="C:" GOTO ERROR
IF "%1"=="c:" GOTO ERROR
GOTO HELP

:ERROR
ECHO You cannot format drive C!!!
GOTO END

:FORMA
FORMAT! %1 %2 %3 %4
GOTO END

:HELP
:CASE64
CLS
ECHO Select the application to run.
ECHO"
ECHO     (A) Format 160K, single-sided, 5 1/4-inch
ECHO     floppy to LOW DENSITY
ECHO     (B) Format 180K, single-sided, 5 1/4-inch
ECHO     floppy to LOW DENSITY
ECHO     (C) Format 320K, double-sided, 5 1/4-inch
ECHO     floppy to LOW DENSITY
ECHO     (D) Format 360K, double-sided, 5 1/4-inch
ECHO     floppy to LOW DENSITY
ECHO     (E) Format 1.2MB, double-sided, 5 1/4-inch
ECHO     floppy to HIGH DENSITY
ECHO     (F) Format 720K, double-sided, 3 1/2-inch
ECHO     floppy to LOW DENSITY
ECHO     (G) Format 1.44MB, double-sided, 3 1/2-
ECHO     inch floppy to LOW DENSITY
ECHO     (H) Format 2.88MB, double-sided, 3 1/2-
ECHO     inch floppy to HIGH DENSITY
ECHO     (I) Exit
GETKEY
SET OPT=64
REM Type the next two lines as
REM one program line.
FOR %%E IN (0 65 66 67 68 69 70 71 72 73 74) DO IF
  ERRORLEVEL=%%E SET OPT=%%E
GOTO CASE%OPT%
:CASE0
ECHO Invalid selection, please try again.
PAUSE
GOTO CASE64
:CASE74
ECHO Invalid selection, please try again.
PAUSE
GOTO CASE64
:CASE65
ECHO Format 160K, single-sided, 5 1/4-inch floppy
ECHO to LOW DENSITY
FORMAT! A: /F:160
GOTO CASE64
:CASE66
ECHO Format 180K, single-sided, 5 1/4-inch floppy
ECHO to LOW DENSITY
FORMAT! A: /F:180
GOTO CASE64
:CASE67
ECHO Format 320K, double-sided, 5 1/4-inch floppy
ECHO to LOW DENSITY
FORMAT! A: /F:320
GOTO CASE64
:CASE68
ECHO Format 360K, double-sided, 5 1/4-inch floppy
ECHO to LOW DENSITY
FORMAT! A: /F:360
GOTO CASE64
:CASE69
ECHO Format 1.2MB, double-sided, 5 1/4-inch floppy
ECHO to HIGH DENSITY
FORMAT! A: /F:1.2
GOTO CASE64
:CASE70
ECHO Format 720K, double-sided, 3 1/2-inch floppy
ECHO to LOW DENSITY
FORMAT! B: /F:720
GOTO CASE64
:CASE71
ECHO Format 1.44MB, double-sided, 3 1/2-inch
ECHO floppy to HIGH DENSITY
FORMAT! B: /F:1.44
GOTO CASE64
:CASE 72
ECHO Format 2.88MB, double-sided, 3 1/2-inch
ECHO floppy to HIGH DENSITY
FORMAT! B: /F:2.88
GOTO CASE64
:CASE73
SET OPT=
CLS
ECHO Exiting Format Menu.
:END
```

End

Q&A DOS, DISKS, AND DILEMMAS

DOS

Q My Leading Edge 286 has a 40MB hard drive with two data partitions (a 32MB partition as drive C, and an 8MB partition as drive D). I bought the system with MS-DOS 3.2 installed several years ago. When I use the FDISK command to request information about my hard disk partitions, the program doesn't report the drive D information. When I type this command:

```
FDISK D:
```

the program provides only a report on my primary drive, C. Am I using FDISK incorrectly? Or is something wrong with my hard drive's data partitions?

A You're using FDISK correctly, and there's likely nothing wrong with your hard drive's data partitions. The FDISK utility supplied with DOS 3.2 did not inherently allow you to create multiple DOS partitions, or partitions larger than 32MB. (DOS did not provide this support until Version 3.3).

To circumvent the DOS 3.x partition limitations, hard drive manufacturers created proprietary disk partitioning software, which they supplied with each high-capacity hard drive. Unfortunately, these programs were not always compatible with the partition table format used by DOS.

Therefore, your version of FDISK may not recognize the drive D partition created by a third-party disk partitioning program. Typically, computer dealers during the DOS 3.2 era used this utility to set up a hard drive before they sold a system, but they'd forget to provide the software to the customer. If you bought your computer used, the previous owner may never have had the software or may have neglected to pass on the special software with the system.

A visit to your local user group or computer dealer may turn up a program that can read the partition table data. However, the easiest way to solve your problem is to upgrade to the latest DOS version.

I suggest you back up the programs and data on your hard drive, then upgrade your system to DOS 5.0. This will let you format your hard drive using a single, large data partition. It also will ensure compatibility with future DOS versions.

—J.W.

Q Is the VERIFY command the same as the /V switch in COPY or XCOPY? Can I place a reference to it in my system's AUTOEXEC.BAT file? Will it verify data stored to disk from my application programs or verify only when I copy files from place to place at the DOS command line? Finally, can I use it as a diagnostic tool to point out bad sectors on my disk?

A The VERIFY command and the /V switch used by the COPY and XCOPY commands invoke the same disk-sector verification process, the purpose of which is to ensure that DOS will not write data to a bad disk sector. The command verifies the integrity of the magnetic area of the disk to which the data will be written; it cannot verify the integrity of the data in a file.

When verification is turned on, DOS performs the same ritual before writing data to each disk sector. It first writes a special pattern of bytes, called a checksum, to the disk, then immediately reads back the pattern. If the read and write byte patterns match, the sector is declared good. This ensures there is no physical damage in the area of the disk where DOS is about to place your data.

Remember to disable disk caching software (such as SMARTDRV.SYS) before you attempt to use VERIFY (or /V). Most disk caching utilities temporarily store information in memory instead of writing it directly to your disk. A disk cache will force VERIFY to write and read its checksum patterns in memory, which will effectively defeat its purpose.

VERIFY (or /V) can be invoked only by DOS commands that write data to disk. Applications that bypass DOS to write data to your disk cannot take advantage of VERIFY. Applications that use DOS commands to write information to disk, however, will use the verification process automatically if you include a VERIFY ON statement as a line item in your AUTOEXEC.BAT file. In this case, you needn't add the /V switch to each DOS command (although no harm will be done if you do). When you type a VERIFY OFF command, DOS will discontinue the verification process.

You can't use VERIFY as a diagnostic tool because it provides no

BY HARDIN BROTHERS ••• JAMES LANGDELL ••• JOHN WOLFSKILL

feedback to the user when it encounters a bad disk sector. The FORMAT command is the only DOS command that can reliably identify bad disk sectors. But you should use FORMAT only on new disks or disks that contain data you wish to erase. Typically, the FORMAT command will spot and "lock out" all bad sectors on a new disk before you use it. VERIFY, by comparison, primarily guards against physical damage that may occur after the disk is formatted.

—J.W.

Q Does DOS's SHARE command have any value for the single user? How can I best take advantage of it?

A An external DOS command, SHARE.EXE is intended primarily for users of local area networks (LANs). Its main purpose is to provide file-sharing and -locking capabilities. File-sharing lets several different users or programs access a data file, while file locking prevents multiple programs from attempting to update a data file at the same time. DOS uses the code loaded by SHARE.EXE to validate all disk read and write requests.

If you use DOS 4.x, and your hard drive contains a single primary data partition larger than 32MB, you should load SHARE.EXE to ensure proper operation of the DOS file-handling system. However, if you use DOS 5.0 or later, there's no need to load the program, unless you use a package such as Windows or DESQView to do multitasking. I know of no other benefits that SHARE.EXE provides.

—J.W.

Q Do I have to use Edlin if I want to add special characters, including the escape character, to a batch file, or can I use DOS 5.0's Edit utility? If Edit works in this instance, is there any reason to keep Edlin on my hard disk?

A Programmers often add special characters to ECHO statements in a batch file for two purposes. They use the so-called extended characters (those with ASCII values ranging from 128 to 255) to create boxes and other special screen effects. The escape character (ASCII 27) lets them send commands to DOS's ANSI.SYS utility to change screen colors, clear parts of the screen, position the cursor, and create keyboard macros.

With DOS 5.0's Edit, you can add any characters you wish to a batch file. Press Ctrl-P, and then press the character you want. For example, to include the escape character in a file, press Ctrl-P, and then press the Esc key. For extended characters, press Ctrl-P, then hold down Alt and type the character's three-digit number on the numeric keypad.

I don't know of any reason to use or keep Edlin on your hard disk. Most word processors let you add special characters to a file, and virtually any word processor or editor is easier to use than Edlin. If you don't have DOS 5.0's Edit utility or a word processor that will work, several easy-to-use shareware text editors are available that will do the trick. One such shareware program that I like is Boxer.

—H.B.

Q I have heard people refer to CHKDSK as a diagnostic program, but I am confused. When should I use it? Does it "fix" anything? Can you list two or three things that I can do with this command?

A CHKDSK tells you the name of a disk; the total amount of space on the disk; the amount of space used by hidden files, directories, and normal files; the number of directories and files on the disk; and the amount of space available. It also tells you how much standard RAM you have and how many bytes are free. You can use it to find any of this information, most of which is not otherwise available.

CHKDSK also checks and, optionally, repairs the directory structure on a disk. It does so by making sure that every file is stored in a unique position on your disk and that every portion of the disk is part of a file, marked as unusable or as free and available for storage of new data.

If CHKDSK finds any storage errors, it asks your permission before fixing the directory structure. If you grant permission, it saves cross-linked data (information that appears to belong to two files at once) and "lost" data (information that doesn't belong to any file). The recovered data is stored in the root directory in files that have a CHK extension. To make these repairs, however, you must start with the following command:

```
CHKDSK /F
```

Once CHKDSK moves the data into the special files, you can use a sector editor (such as the one included with PC Tools or The Norton Utilities) to examine it and move anything you want to keep into a new file. Then you can delete the CHK files to recover your hard disk space.

You should use CHKDSK on a regular basis to look for and repair storage errors and to ensure your data is stored correctly. Many people run CHKDSK from their AUTOEXEC.BAT file each time they boot their computers to make sure their hard disks are in good shape. You also should use it any time a program ends abnormally—if the program hangs and you have to reboot, for instance—or if your computer loses power while an application is running.

—H.B.

Q What kind of limit does DOS's syntax for subdirectories impose? Does this limit apply to the number of directories or the number of characters? I am thinking about the problems that might occur if I rename a subdirectory in a long pathname and in so doing end up exceeding DOS's limit. Would I lose access to files at the end of the chain?

A The limit on the length of pathnames is 63 characters, including the backslashes that are part of the name. (If you count the initial slash that indicates the root directory, the longest possible name is 64 characters.) Once you create a directory within those limits, you needn't worry about the length of filenames within that directory, even if the full directory path plus the filename totals more than 63 characters.

It's possible to use the MKDIR (MD) command to create a directory that exceeds this character limit. But you can't use the CHDIR (CD) command to move into that directory or use COPY to move files there. Fortunately, the RMDIR (RD) command lets you remove the useless, too-deeply nested directory.

DOS imposes no limit on the number of directory levels you can create, as long as the pathname contains no more than 63 characters. By giving each directory level a name only one character long, you could create a directory 32 levels deep. If, for some reason, you rename a directory and exceed those limits, you can always recover by giving the directory a shorter name.

—J.L.

Q I have heard a great deal about MS-DOS 5.0 but not so much about the latest version of IBM PC-DOS. The computer magazines used to make much more out of the distinctions. Are they now the same? What are the differences? If you had a choice, would you run PC-DOS on a non-IBM computer? Would you run it even on an IBM computer?

A Up through Version 2.0 of DOS, Microsoft let each vendor make internal modifications to the operating system. Every version of MS-DOS could be different in subtle, but important, ways. The only well-known version was IBM's (because it was the most widespread), so many people used PC-DOS even on non-IBM computer systems.

Unfortunately, this sometimes created problems for users who were interested in programming in Basic, because every IBM computer has additional copyrighted code in the ROM that contains part of Basic. The Basicprogramming language included with PC-DOS runs only on computers made by IBM and locks up on all other computers. The MS-DOS version of Basic (usually called GW-Basic) runs on all DOS computers, including IBMs.

Beginning with Version 3.0, Microsoft maintains complete control over MS-DOS. Vendors no longer can change the internal code, and MS-DOS is a known entity. Also, minor operational differences between IBM's and Microsoft's versions of DOS have disappeared.

I have not tried IBM's version of DOS 5.0. I have heard that its QBasic and EDIT.COM commands use the special ROM code and therefore, will run only on IBM computers. But I see no reason to use PC-DOS on a non-IBM computer.

If I used an IBM computer, I would likely use MS-DOS anyway, because it has a lower street price (a higher volume of sales brings a lower cost) and because it is now the best-known version. I have not heard of any reason, however, to avoid using PC-DOS 5.0 on an IBM computer.

—H.B.

Q I want to use the SUBST command to create a directory as a separate drive. I hope this will let me shorten my PATH statement. But I'm not sure I'm doing it correctly. What is the appropriate syntax for renaming, let's say, PC Write as a unique drive? How do I refer to PC Write thereafter—by its drive designation, its name, or both?

A SUBST is the correct command for creating an artificial drive letter as a name for a real directory. (Its companion command, JOIN, assigns an artificial directory name to a real drive.)

First, you should make sure that the drive letter you're trying to assign is one that DOS is ready to recognize. To establish drive letters beyond those your actual drives and hard disk partitions use, include a

LASTDRIVE command in your CONFIG.SYS file. For example, if you want drive letters that ran through the letter *J*, include the following command in your CONFIG.SYS file:

```
LASTDRIVE=J
```

For each additional drive letter you request, DOS will consume about 80 bytes of memory, so there's a mild penalty for requesting more drive letters than you'll actually use.

Now, you can write a SUBST command line that uses one of those drive letters. Say you had your PC Write program files in a directory on drive C, named \PROGRAMS\PCWRITE. To establish drive G as a name for this directory, you would enter this command:

```
SUBST G: C:\PROGRAMS\PCWRITE
```

If you're making this change to use a shorter name in the PATH command line, be sure the command is executed *before* you run the PATH command that mentions drive G. And, if you include your PATH command in your AUTOEXEC.BAT file, be sure the SUBST command line comes before the PATH command in that file.

With the SUBST command in effect, you can use the new drive name or the actual pathname of PC Write's directory. Certain commands will disregard drive letters established by SUBST, but these are not commands that would involve a program directory in normal operations.

If you're attempting to use SUBST to evade the 127-character limit in the PATH command, other solutions are available. For example, you can use the APPEND command with its /X option to provide an additional (or appended) list of directories to be searched for program files. (Be careful, however, APPEND can wreak havoc if it is used improperly.) If you regularly set up directories for data files beneath the directory that runs each program, here's another possible way to shrink the PATH command line. You can include a double period entry (..) to tell PATH in general terms to search the directory immediately above the current directory. For example, you can insert the following line in your AUTOEXEC.BAT file:

```
PATH C:\;C:\DOS;..;C:\UTILITY
```

—J.L.

Q I want to trade up to DOS 5.0, but I have many back up files created by DOS 3.2. Should I restore all my files first and then upgrade to DOS 5.0?

A You do well to question whether you can use your backup files under a new DOS version. In years past, the BACKUP and RESTORE commands changed greatly from one version of DOS to the next, and later versions of RESTORE couldn't handle files archived under earlier versions of BACKUP.

Fortunately for you, those commands were mature enough by the emergence DOS 3.2 that all later versions of RESTORE can handle files archived with DOS 3.2's BACKUP command. Earlier DOS versions, including 3.1, would cause problems, however.

Users who have a copy of DOS prior to 3.2 need to take the following special steps to reach compatibility between their old and new versions. First, boot your computer with the earlier DOS version. Then use that DOS's RESTORE command to reconstitute the desired files into their standard file format. Keep those files in that standard form until you're using them within the newer DOS version. Once you are running DOS 3.2 or later, you can use that version's BACKUP command to save the files in a form a later DOS version can restore.

—J.L.

BATCH FILES

Q Can I get a mouse to work in a batch file? And can I do it without listing the mouse in my CONFIG.SYS file?

A If you want to use a mouse with a batch file menu, you probably can do so with the help of a utility program, but the task requires some special programming with Debug or another programming language.

If you simply want to activate the mouse for use in an application, however, your task is easy. Almost all mice are shipped with two programs that have names such as MOUSE.SYS and MOUSE.COM. You can use the first of these programs in your CONFIG.SYS file to enable the mouse for all application programs.

The second program, MOUSE.COM, can enable your mouse for all applications from your AUTOEXEC.BAT file, or it can turn on the mouse for a particular application and then disable it when the program ends. To turn off the mouse after an application program has run, the version of MOUSE.COM you use must support a disable command; not all versions of MOUSE.COM do so.

If you want to activate the mouse for a single application—perhaps because you need every available megabyte of memory to run a program that isn't mouse-compatible—you can write a batch file that looks something like this:

```
MOUSE ON
program name
MOUSE OFF
```

See your mouse documentation to find out the correct program name to use.

—H.B.

Q I'm just getting up to speed with batch files and don't have all the tricks down yet. Can you tell me if it's possible to add sounds to a batch file?

A If you want a simple beep or two, you can echo a Ctrl-G (ASCII character 7) to the screen using the ECHO command. If you want more complex sounds, you will need a utility program, such as the Batch Enhancer program included in The Norton Utilities, to create the sounds. You then can call the program as needed from your batch files.

—H.B.

HARDWARE

Q If files are not saved contiguously on a disk, how does DOS find all the parts of a file? I know it employs a file allocation table, but is it correct to assume that the FAT maintains a record of every part of every file, or just its beginning?

A The FAT is a large data table that maintains the disk cluster locations assigned to all parts of a file. Each 2-byte (16-bit) entry in the table represents the location of a disk cluster assigned to a particular file.

When DOS assigns space to create or extend a file, it allocates space in increments called disk clusters. For example, the data platter of a typical 40MB hard drive contains thousands of disk clusters, each comprised of four 512-byte disk sectors.

The best way to describe how DOS uses the FAT to find all parts of a file is to compare the process to a treasure hunt. Clue 1 leads you to clue 2, which leads you to clue 3 and so on until you reach the treasure. Similarly, a file's directory entry contains only the starting number of the entire FAT entry, which represents the physical starting cluster for the file. For example, a file's directory entry may specify FAT entry 2 as the starting point. (This also indicates that disk cluster 2 holds the first portion of the file.) When DOS reads the 16-bit value stored in this FAT entry, it points to, or references, the FAT entry containing the next portion of the file.

For example, the 16-bit number stored in FAT entry 2 may be 00 03. This number indicates the next cluster occupied by the file is disk cluster 3. DOS then proceeds to read the value stored in that FAT entry. The process continues in this manner until DOS reaches a FAT entry that contains the special value FF FF, which indicates the entry is the last cluster occupied by the file. This scheme forms the file's logical "FAT chain."

DOS follows the links of the chain until it locates all clusters that belong to a particular file. Admittedly, the process is a bit abstract, but it's also very efficient. The FAT cluster chain allows DOS to quickly allocate and release disk space without constantly shuffling the data table entries.

—J.W.

Q I've decided to update to DOS 5.0. My IBM XT's 42MB hard drive (with DOS 3.3 installed) includes two data partitions; a 20MB partition as drive C, and a 22MB partition as drive D. What will happen when I try to create one large data partition instead of two smaller ones? Will DOS 5.0 know that my hard drive has two partitions? Will I risk losing all my programs and data files?

A When you run its installation program, DOS 5.0 will recognize that your hard drive was formatted with two DOS partitions (typically, a primary DOS partition as drive C, and an extended DOS partition as drive D). It also will see that the partitions contain data. Accordingly, it will warn you that all data will be lost if you attempt to create a single data partition. At this point, you'll be provided with an opportunity to exit the Setup program.

Therefore, before you begin the DOS 5.0 installation, make sure you first back up your files. If you don't own a third-party disk backup program, you can back up your hard drive by installing DOS 5.0 as though your system contained only floppy disks. After you complete the floppy disk installation, place the boot disk in drive A, then restart your system. Now you can use the DOS 5.0 utility BACKUP.EXE to back up your hard drive's files to floppy disks. After the backup is complete, install DOS 5.0 on your hard drive, then use RESTORE.EXE to place the files back on your hard drive. ■

—J.W.

SUBSCRIBE
TODAY
DOS RESOURCE GUIDE
800-238-6580

PC Essentials For Everyday Users
DOS RESOURCE GUIDE
The PC Productivity Magazine
Make DOS Work For YOU!
More Than 140 TOP Programs Reviewed
GUIDE TO WINDOWS SOFTWARE
COVERS ALL DOS VERSIONS
PRESENTS: THE ESSENTIAL GUIDE TO DOS
2nd Edition
Batch File Tips and Tricks
MANAGING MEMORY STEP-BY-STEP
COMPLETE GUIDE To DOS Files

AND SAVE 20% OFF

DOS RESOURCE GUIDE
The PC Productivity Magazine
THE COMPLETE DOS Q&A:
GET STARTED WITH QBASIC
HELP US OUT: READER SURVEY INSIDE...PAGE 81
THE ONE-STOP DOS PRIMER
The 10 Worst DOS Commands
DEBUG Almost Defined
15 DOS TIPS
PROTECT YOUR DATA FROM VIRUS ATTACK

THE COVER PRICE

SUBSCRIBE TODAY
DOS RESOURCE GUIDE
800-238-6580

PC Essentials For Everyday Users
DOS RESOURCE GUIDE
The PC Productivity Magazine
Make DOS Work For YOU!
Speed Up Your Hard D
More Than 140 TOP Programs Reviewed
GUIDE TO WINDOWS SOFTWARE
COVERS ALL DOS VERSIONS
PRESENTS: THE ESSENTIAL GUIDE TO DOS
2nd Edition
NOW INCLUDES DOS 5.0
Batch File Tips and Tricks
MANAGING MEMORY STEP-BY-STEP
COMPLETE GUIDE To DOS Files
DOS 5.0

AND SAVE 20% OFF

THE COVER PRICE

Give DOS 5.0 A Friendlier Face

You can brighten up your dull DOS world and make it better suit your needs simply by adjusting DOS 5.0's initialization file.

by Jack Nimersheim

Anyone who has purchased Windows is familiar with INI files. Windows relies on the information stored in these files to prepare, or initialize, itself and other programs to run properly on a given hardware setup (hence, the INI extension by which the files are identified). What you may not know is that the newest version of DOS also includes an INI file. MS-DOS 5.0 uses its INI file, DOSSHELL.INI, to control specific elements of the DOS shell environment.

Although DOSSHELL.INI is not nearly as comprehensive as Windows' INI files, it performs a few useful functions. For example, DOSSHELL.INI specifies the color scheme you select for your DOS shell environment, when you use the Color option in the pull-down Options menu. It also specifies the combination of colors that defines a particular color scheme. The display mode (graphics or text) and resolution are two other examples of the types of settings stored in DOS's INI file.

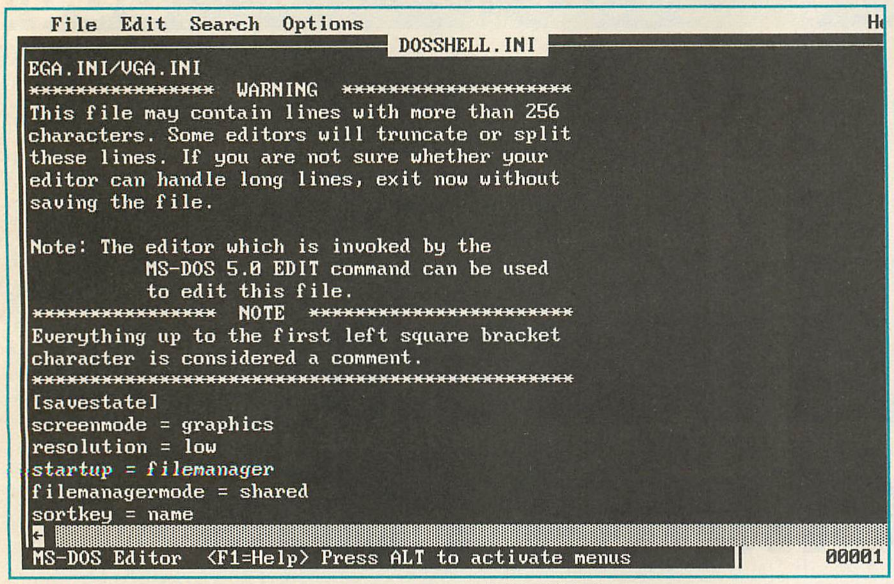
Normally, the contents of DOSSHELL.INI are changed automatically whenever you select a menu option that modifies your DOS shell. But, because DOSSHELL.INI is an ASCII (standard text) file, you also can change it manually, using almost any word processor or text editor. In this article, I examine the contents of DOSSHELL.INI and show how you can use the EDIT.COM program shipped with DOS 5.0 to modify its contents. After learning a few tips, you'll be able to paint your screen with a splash of bold colors, and, if you're a southpaw, you'll discover how to make your mouse respond better to your needs.

Note: Although DOS 4.x includes a DOS shell, it is not as flexible as the one in DOS 5.0. Nor does this earlier DOS shell use an INI text file to record environment settings. So, you can perform the following exercises only if you have upgraded to DOS 5.0.

Peering Inside the Shell

Before we explore ways to change the DOSSHELL.INI file, you should know a thing or two about how it works. The file is divided into two main sections, called savestate and programstarter. The first section, savestate, contains a number of items that influence the overall appearance and performance of your DOS shell. It contains the settings that determine the display mode (text or graphics) and resolution, which elements of the DOS shell environment are included in your initial display, the order in which files appear in your directory listings, whether these listings should include hidden files, whether or not the DOS 5.0 task swapper is active when you start the DOS shell, and so forth.

For the shell to work properly, each of these settings must have a specific, predefined value. As a rule, then, you should use a text editor to modify these settings only when necessary. Later, I'll provide an example of where you might want to edit the savestate settings manually. Before doing so, however, let me briefly describe the contents of programstarter, DOSSHELL.INI's



```
EGA.INI/VGA.INI
****************  WARNING  ********************
This file may contain lines with more than 256
characters. Some editors will truncate or split
these lines. If you are not sure whether your
editor can handle long lines, exit now without
saving the file.

Note: The editor which is invoked by the
          MS-DOS 5.0 EDIT command can be used
          to edit this file.
****************  NOTE  ***********************
Everything up to the first left square bracket
character is considered a comment.
***********************************************
[savestate]
screenmode = graphics
resolution = low
startup = filemanager
filemanagermode = shared
sortkey = name
```

Photo 1. The comment line [savestate] marks the beginning of DOSSHELL.INI's savestate section, the area that affects the overall appearance and performance of your DOS shell environment.

second and more malleable section.

The first part of the program-starter section contains information about the individual program groups and items in your DOS shell environment. These are the automated routines represented by icons in the DOS shell's Main display window.

During installation, for example, DOS automatically creates a program group called Disk Utilities. You can use individual items in this group to copy or format floppy disks, back up and restore hard disk files, or "undelete" accidentally erased files.

Information recorded in the programstarter section tells the DOS shell what to do when any of these items is selected. Additional program-group icons placed in the DOS shell's Main window by the DOS 5.0 installation program include Command Prompt, Editor, and Microsoft QBasic.

Immediately following the program-group and item definitions is a long section of DOSSHELL.INI. DOS uses this section to identify the color combinations that make up the various color schemes available in the DOS shell environment. The specific variety of color schemes this section contains depends on your type of monitor and the video card you installed when configuring your copy of DOS. Some of the available options include Basic Blue, Ocean, Monochrome, Emerald City, and Hot Pink (one of my favorites).

The final section of DOSSHELL.INI keeps track of the associations you create between data files and applications, when using the Associate option of the DOS shell's File menu. Setting up associations lets you automatically load both a data file and the application program it requires by double-clicking on the data file's name in the DOS shell directory listing.

In fact, creating a new association seems a good place to illustrate how DOS 5.0 uses the DOSSHELL.INI file. I'll show you how to do this first, then I'll describe the specific steps required to edit the INI file.

Digging for DOSSHELL.INI

Before you begin, you should create a copy of the current DOSSHELL.INI file. This ensures you'll be able to restore your initial DOS shell environment if any problems arise while editing DOSSHELL.INI.

Follow these steps to make a copy of DOSSHELL.INI:

- Press Tab until the Directory Tree is the active window in the display of your DOS shell environment.
- Use the arrow keys to highlight the DOS directory.
- Press Tab to make the file listing the active windows in your DOS shell display.
- Use the arrow keys to highlight the DOSSHELL.INI file.
- Press Alt-F to display the pull-down File menu.
- Press *C* to select Copy.

When the DOS shell displays its Copy File dialog box, type:

```
C:\DOS\DOSSHELL.HLD
```

and press Enter. This creates a duplicate copy of your original DOSSHELL.INI file, called DOSSHELL.HLD.

Next, use the Associate option to tell the DOS shell to run the DOS 5.0 text editor, EDIT.COM, whenever you select DOSSHELL.INI from the DOS shell file listing. To do this:

- Use the arrow keys to highlight the EDIT.COM file.
- Press Alt-F to display the pull-down File menu.
- Press *A* to select Associate.

When the DOS shell displays its Associate File dialog box, type the three-letter extension *INI*. Then, press Enter. This creates a link between EDIT.COM and any INI files.

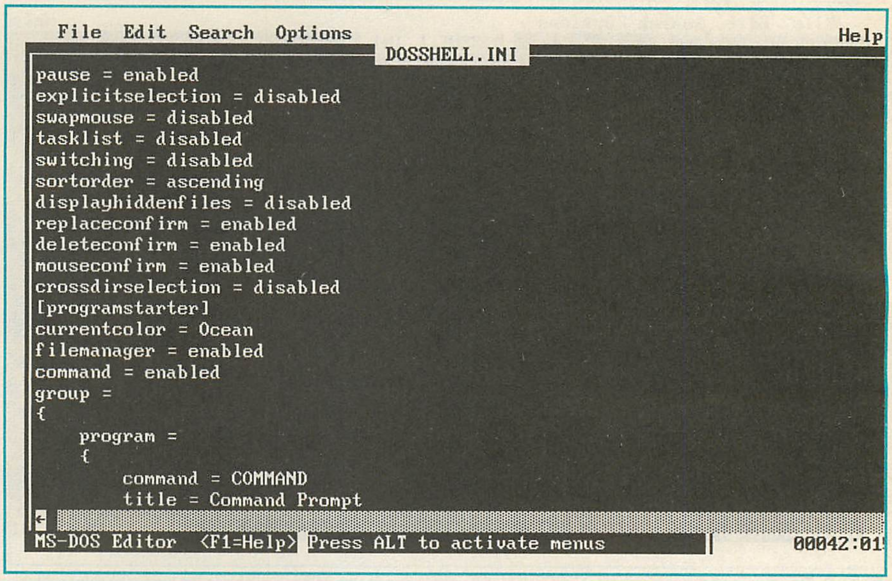
Having laid this groundwork, you're ready to use the DOS 5.0 text editor to dig into the DOSSHELL.INI file. You start by opening DOSSHELL.INI for editing. Then, use the arrow keys to highlight the DOSSHELL.INI file, and press Enter.

This procedure runs EDIT.COM and automatically loads the DOSSHELL.INI file for editing, as illustrated in Photo 1. Notice that about halfway down your screen, the word *savestate* is enclosed in brackets. As the note immediately above it explains, this is a comment line used to identify the beginning of the savestate section of the DOSSHELL.INI file (which I described earlier). The first two lines in the savestate section of Photo 1 indicate that, on my system, the DOS shell is set up to run in graphics mode, at low resolution.

Note: The specific contents of your DOSSHELL.INI file may differ from that shown in Photo 1, depending on the type of monitor and video adapter you own and how you have configured your DOS shell environment.

As I stated earlier, you rarely will want to change the settings in the savestate section of your DOSSHELL.INI file. In fact, doing so could cause problems, given that specific values must be associated with each setting for the DOS shell to work properly. Still, curiosity is a powerful emotion, so I've described the functions of a few of the settings in the table, "Savestate Settings."

The second section of savestate contains the item I alluded to earlier—the rare case in which you might want to edit savestate settings. Here, left-handed individuals may want to turn on the swapmouse option. Doing so lets them use the right mouse button to select items in the DOS shell display. You must use a text editor to make this manual modification to DOSSHELL.INI.



```
pause = enabled
explicitselection = disabled
swapmouse = disabled
tasklist = disabled
switching = disabled
sortorder = ascending
displayhiddenfiles = disabled
replaceconfirm = enabled
deleteconfirm = enabled
mouseconfirm = enabled
crossdirselection = disabled
[programstarter]
currentcolor = Ocean
filemanager = enabled
command = enabled
group =
{
    program =
    {
        command = COMMAND
        title = Command Prompt
```

Photo 2. The second screen display of the savestate section lets you see whether you have activated a task list, whether you have multiple programs running under the DOS shell, and whether files in your directory listings will be sorted in ascending or descending order.

Note: Turning on the swapmouse setting disables the left mouse button. Don't perform the following exercise unless you want to use the right mouse button exclusively to select items from your DOS shell display.

To enable the swapmouse option:

- Use the arrow keys to position the on-screen cursor under the letter *d* of the disabled setting in the line that begins with the word *swapmouse*.
- Press Ctrl-T to erase the original setting.
- Type the word *enabled*.

Later, when you save the modified DOSSHELL.INI file to disk and return to the DOS shell, this new setting will activate the right mouse button—a much more convenient arrangement for left-handed users.

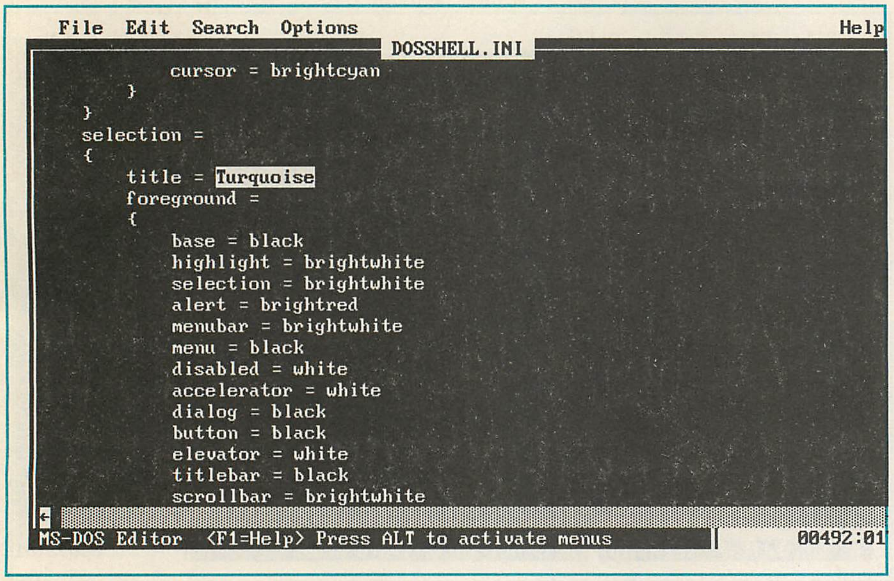
SAVESTATE SETTINGS

screenmode	This line identifies the display mode the DOS shell uses. Valid options are text and graphics.
resolution	When the DOS shell is configured to run in graphics mode, this line is used to select a display resolution. Valid options differ, depending on your type of monitor.
sortkey	This line identifies the order in which your file listings are displayed. The name setting in Photo 1 indicates that my listings are sorted in alphabetical order, based on the filenames. Other valid settings for sortkey include extension, date, size, and diskorder. Because DOSSHELL.INI is too large to fit into a single display screen, you must press the PgDn key to view the next section. Your screen now should resemble Photo 2, which shows the remaining lines in the savestate section. Items in this second screen of settings include tasklist, switching, and sortorder.
tasklist	The setting in this line indicates whether you have activated DOS 5.0's Task List.
switching	The setting in this line indicates where task swapping is currently active—that is, whether you have multiple programs running under the DOS shell.
sortorder	This setting specifies whether files in your directory listings will be sorted in ascending or descending order.

Color Me Not-So-Basic Blue

Approximately halfway down the current screen, you'll see the word *programstarter* enclosed in brackets (see Photo 2). This line indicates the beginning of the programstarter section.

The first line in this section, the currentcolor setting, indicates the color scheme you've selected for the



```
            cursor = brightcyan
        }
    }
    selection =
    {
        title = Turquoise
        foreground =
        {
            base = black
            highlight = brightwhite
            selection = brightwhite
            alert = brightred
            menubar = brightwhite
            menu = black
            disabled = white
            accelerator = white
            dialog = black
            button = black
            elevator = white
            titlebar = black
            scrollbar = brightwhite
MS-DOS Editor  <F1=Help> Press ALT to activate menus                00492:01
```

Photo 3. This screen shows the section of DOSSHELL.INI used to specify the colors available in the turquoise color scheme.

DOS shell. DOS 5.0's shell provides an attractive alternative to the admittedly boring DOS system prompt. In all honesty, however, the various colors available in the DOS shell's default color schemes are not that exciting. To spice them up, try the following experiment:

- Use the arrow keys to position the on-screen cursor under the first letter of the currentcolor setting.
- Press Ctrl-T to erase the current setting.
- Type the word *turquoise*.

This changes the active color scheme to turquoise. (You should note, however, if the color you select is not available in your hardware configuration, DOS will display the default hue, a shade of gray.) Next:

- Press Alt-S to display the pull-down Search menu.
- Press *F* to select Find.
- When the DOS shell displays its Find dialog box, *turquoise* already will be entered in the Find What field. To move to the next occurrence of this word in the DOSSHELL.INI file, press Enter.

Your screen now should resemble Photo 3, which shows the section of DOSSHELL.INI used to specify the colors available in the turquoise color scheme. This color scheme normally consists of several different shades of blue—not unattractive, to be sure, but not provocative, either. So, let's liven it up.

To modify the turquoise color scheme:

- Press PgDn to display the background settings for the turquoise color scheme.
- Use the arrow keys to position the on-screen cursor under the first letter in the command that reads: `base = brightwhite`
- Press Ctrl-T to erase the current setting.
- Type *brightmagenta*.
- Use the arrow keys to position the on-screen cursor under the first letter in the command that reads: `menu = brightcyan`
- Press Ctrl-T to erase the current setting.
- Type *brown*.
- Use the arrow keys to position the on-screen cursor under the first letter in the command that reads: `titlebar = white`.
- Press Ctrl-T to erase the current setting.
- Type *brightgreen*.

Before saving your new settings, press PgDn. You now should be looking at the DOSSHELL.INI section that handles associations (look for the word *associations* enclosed in brackets). Notice the two lines specifying the program and extension settings for EDIT.COM. Earlier, the DOS shell automatically inserted this section into DOSSHELL.INI when you used the Associate option to create a link between EDIT.COM and files ending with an INI extension. As you can see, the DOSSHELL.INI file is updated constantly as you make changes to your DOS shell environment.

Speaking of changes, let's save your modified DOSSHELL.INI file and see the results. To save it and return to the DOS shell:

- Press Alt-F to display the pull-down File menu.
- Press *S* to select Save.
- Press Alt-F to display the pull-down File menu.
- Press *X* to select Exit.

This closes EDIT.COM and displays a "Press any key to return to the MS-DOS Shell..." message. Go ahead and press any key. When you return to the DOS shell, you'll be greeted by a splash of tints and hues unavailable in any of its default color schemes. If this new color combination isn't your cup of tea, don't worry. When you copy back to DOSSHELL.INI the DOSSHELL.HLD file you created earlier, the original turquoise color scheme is restored.

Those familiar with Windows and its various applications could tell you that DOSSHELL.INI clearly is not as flexible as its Windows counterpart. That's understandable; the DOS shell itself wasn't designed to be a true graphical user interface (GUI), as was Windows. Its primary function is to put a friendlier face on DOS 5.0—an undertaking at which it succeeds admirably. ■

Computer journalist and science-fiction writer Jack Nimersheim has been using computers for more than 10 years. During that time, he has published 13 books and some 450 articles on computer-related topics.

Debug Does Disks

Sooner or later, you'll have to use DOS's quirkiest power tool on your delicate disks. Here's how to make Debug read and write without too much 'rithmetic.

by Dan Gookin

Debug's reputation as a lethal instrument comes mostly from its potential to turn a slip of the finger into a corrupted floppy-disk file or, worse, a trashed hard disk. We steered clear of such perils in the last installment by peeking and poking data into your PC's memory addresses, avoiding disks altogether. It was a painless exercise, harmless (for the most part) and fun—not at all the cryptic and scary Debug the DOS elite claim it to be.

This time, we'll move on to more advanced and potentially dangerous uses—namely, commanding Debug to read from and write to disk.

Debug offers three different ways to perform disk functions. While running Debug, you can load a file from disk into memory, or you can load chunks of raw data from disk, whether or not that data is in a file. Once the data's in memory, you can use Debug's memory commands to manipulate it. From there, if you like, you can write the data back to disk. Outside Debug, you can write Debug scripts in an ASCII text editor or word processor, then tell DOS to feed the resulting file into Debug to create an executable COM file that resides on disk. (Refer to the table, "Debug's Disk Commands," for a summary of the six commands involved.)

Great, you're saying to yourself: Why on earth would I even consider doing that? Well, in its arcane way, Debug's disk functions let you do a number of nifty things, such as change prompts and messages in binary files that can't be accessed easily any other way. You also can use it to piece together corrupted text files that have become unreadable through normal means. (This method isn't practical with binary, or nontext, files, because it's nearly impossible to visually scan for the separated program segments.) To be honest, since Debug's introduction, a number of commercial packages have come along (PC Tools and The Norton Utilities, for example) that can perform such acts of prowess, not to mention binary-data recovery and disk repair, with greater ease.

Realistically, the instructions that follow will be most useful in helping you manage the many Debug files you'll undoubtedly be inspired to create once Debug grabs you. After all, doing all your work in memory leaves nothing permanent to show for your terribly brilliant efforts.

The Dawn of Creation

Start Debug and prepare to create a tiny text file by typing `C:\DOS\DEBUG` (assuming that DEBUG.COM resides in the DOS directory on drive C). As usual, the friendly hyphen prompt greets you. You're now going to use the E command to poke a few lines of text into memory. Memory location 100 hexadecimal is fine for this exercise; Debug automatically assumes 100

DEBUG.COM: A set of DOS commands originally designed to assist programmers in finding and patching programs.

PROGRAM TYPE: External.

AVAILABLE: MS-DOS 1.25 and later. Updated in DOS 2.0 and again in DOS 4.0.

DEBUG'S DISK COMMANDS

D Displays data currently residing in memory. Handy for double-checking data before saving it to disk. Also displays disk data by sector after it is loaded into memory.

Examples:

```
D100
```

Displays 128 bytes of memory (a default value, automatically assumed by Debug) starting at 100 hex.

```
D100 L200
```

Displays memory contents starting at 100 hex and showing 200 hex (512) bytes, or one sector's worth, of disk data.

E Enters data in memory for later saving to disk.

Example:

```
E100 "This is a sample file" OD OA
```

Loads the brief message at memory location 100 hex, following it with a carriage return and line feed (0D and 0A commands).

L Loads a file from disk or loads disk data by sector (ignoring filenames).

Examples:

```
L
```

Loads a previously named file using the N command.

```
L100 0 0 1
```

Loads sector-specific disk data into memory, starting at 100 hex. First zero specifies drive A, second zero means first disk sector, and 1 designates the number of sectors to be loaded.

hex each time it starts. At the hyphen, type in the following:

```
E100 "This is a sample file" OD OA
```

That's the E command followed by 100 hex, the location where you can start poking bytes. You're poking in the line "This is a sample file", which must be enclosed in double quotes. It's followed by two single-byte values: 0D, the carriage return, and 0A, the line feed.

After pressing Enter, nothing much happens. However, you can use the D command to dump memory and see the text you've poked into memory:

```
D100
```

The D command displays 128 bytes of memory. In the far right (ASCII) column, you'll see your text, starting on the first line and wrapping over to part of the second. It ends at memory location offset 116, where you can see the value 0A in the seventh column (the columns start with a zero).

You'll now enter a second line of text, which should start at location 117, the byte after 0A. (This is complex, sure, which is why Debug isn't renowned for editing text.) Type in the following:

```
E117 "created by Debug." 1A
```

Again, you have the E command, this time poking in text at memory location 117. The last byte is 1A, which is the Ctrl-Z character that ends every text file under DOS.

Use the D command again to display the text:

```
D100
```

Everything should appear just as you entered it.

Writing Skills for Beginners

Debug can write any chunk of memory either directly to disk or to a file. Since writing to a file is safer, you'll be doing that first. Two things are required: the name of the file and the number of bytes to write. To name the file, use Debug's N command and a filename, such as TEST.TXT:

```
N TEST.TXT
```

You won't notice any feedback after pressing Enter; another hyphen prompt appears. The next step is to tell Debug how many bytes to write to disk. This is a bit tricky to figure.

The 1A character marking the end of the file is at offset 128 hex in memory, as seen by the D command (remember, you should start counting with zero at the right column). When you write a file to disk, Debug assumes it starts at offset 100 hex. Since the file ends at offset 128 hex, the length of the file is 28 hex bytes: 128 - 100 = 28 (and 28 hex equals 40 decimal).

To tell Debug the size of the file, you use the RCX command, which manipulates one of the CPU's registers, a subject I'll get into in the next installment of this series. For now, type the RCX command, and Debug will respond with `CX 0000` and a colon prompt. At the prompt, type 28. The entire sequence will look like this on screen:

N	Names a preexisting or new file. All file-specific Debug operations must be preceded by the N command.	**Example:** `N TEST.TXT` Names a new or existing file to be loaded.

RCX	Tells Debug the size of the file to be saved (written) by loading a hex number into the CPU's CX register. The number equals the file size in bytes.	**Example:**
		`RCX`
		`CX 0000`
		`:28`
		Displays the contents of CX register and accepts hex input (after colon) to change contents to 28 hex.

W	Writes (saves) a file to disk using same syntax as the L command.	**Examples:**
		`W`
		Writes a previously named file using the N command.
		`W100 0 0 1`
		Writes sector-specific data from memory back to disk, reading forward from memory location 100 hex. First zero specifies drive A, second zero means first disk sector, and 1 designates the number of sectors to be written.

```
-RCX
CX 0000
:28
```

This stuffs 28 hex into the CX register, telling Debug how much memory to write.

The final step is easy: Type the W command to write the file to disk, and Debug responds with:

```
Writing 00028 bytes
-
```

Type *Q* to exit Debug. At the DOS prompt, use the DIR command to locate the TEST.TXT file. Note that its size is 40 bytes (in decimal, and thank goodness). Then display your file by typing:

```
TYPE TEST.TXT
```

and you should see:

```
This is a sample file
created by Debug.
```

Learning How to Read

You also can use Debug to read a file from disk. You can do it in two different ways. The first way is to start Debug and specify a filename—just as you would with a word processor or another program:

```
DEBUG TEST.TXT
```

Once in Debug, you can use the D command to view the file in memory. It's stored at location 100 hex. You can use the E command to enter new bytes or the M command to copy bytes. If you want to mess around with it, feel free. You can even use the RCX and W commands to write the file back to disk (remember to specify the proper size for the file). But rather than mess with it, quit Debug: Type *Q* and press Enter.

This first method of loading a file is preferred by programmers who use Debug to test-drive programs. An alternate method allows you a bit more flexibility.

Start Debug again, this time without typing a filename. Once the hyphen prompt appears, you can use the N and L commands to load any file from disk. Start by using the N command to name the file:

```
N TEST.TXT
```

Then type *L* to load it. Don't expect any feedback unless you named a file that doesn't exist (in which case, "File Not Found" pops up). To prove that the file is in memory, use the D command to display it.

This method of loading a file is more flexible, because you can load the file at any place in memory (the L command has an option for specifying a memory location for the file). Nothing was specified above, so Debug stuck your file at memory location 100 hex. But to load it at 128 hex, for example, you would type:

```
L 128
```

For the next demonstration, use the Q command to quit Debug and return to DOS. Then, type the DIR command to put something on the screen. Next, return to Debug and type the following commands:

```
N SCREEN.TXT
RCX
```

When Debug issues a colon prompt, type:

```
FA0
W B800:0
```

and Debug responds with "Writing 00FA0 bytes." The entire sequence should look like this:

```
-N SCREEN.TXT
-RCX
CX 0000
:FA0
-W B800:0
```

Note: If you have a monochrome display, use B000 instead of B800 above, as well as in the next set of commands.

Type *Q* to quit Debug. At the DOS prompt, type the CLS command to clear the screen, then restart Debug. At the hyphen prompt, repeat the above sequence, adding an L command after the line that reads `:FA0` and substituting:

```
M100 FA0 B800:0
```

for `W B800:0`. With the first set of commands, you saved all of video memory—FA0 hex bytes—in a file named SCREEN.TXT. The second time around, you loaded those bytes back into memory using the N, RCX, and L commands. This puts SCREEN.TXT at memory location 100 hex. (Debug won't let you load it directly into video memory.) The M command then is used to copy the data up to video memory: The M100 command tells Debug to start at location 100 hex; FA0 means there are 4,000 bytes to move (copy, actually); and B800:0 is the address of screen memory.

Reading in the Raw

The Debug concepts we'll examine next are not designed for newcomers to DOS or for the faint-of-heart. Using Debug incorrectly while working with disk sectors can be downright deleterious to your system: You can erase or corrupt the contents of a file, make disk files unusable, or even insert instructions that could damage your hardware. So, before you pass Go, be sure to back up all your files. Next, be certain you're working with *copies* of your files, never the originals. Double- and triple-check every command before you execute it. And, finally, if at any time you become uncomfortable about digging into the innards of your PC, type *Q*, then press Enter to quit.

This is the scariest aspect of Debug: using the W and L commands to write and load raw information to and from disk. This technique totally bypasses the DOS file system, which means it's possible to munge a perfectly good disk if you're not careful. Because of that, the following exercises are done on a floppy disk in drive A. My advice is to use an older or "test" disk, specifically one that isn't a DOS boot disk. (If you follow directions properly, nothing will be damaged.)

Quit Debug and restart it. Place your test floppy in drive A, then type the following:

```
L100 0 0 1
```

This command loads information from disk to address 100 hex. The first zero means drive A. The second zero means the first sector of the disk—the boot sector, which identifies the disk as a valid DOS disk. The final value is the number of sectors to load. Above, you're loading only one sector.

To edit information on disk, you first must load it into mem-

TIP FOR YOUR EYES ONLY

A simple, but effective, way to discourage "snoopers" from viewing the programs and data files you've stored on your hard drive is to include a nonprinting ASCII character as part of each filename.

Although DOS lets you enter only ASCII characters 32–127 using the regular typing keys, you can enter the complete ASCII character set (values zero through 255) via your computer's numeric keypad. To enter an ASCII character from the numeric keypad, hold down the Alt key, type the digits that represent the ASCII character you wish to display, then release Alt.

A good choice for a "secret" filename character is ASCII 255, because DOS prints this character as a blank space. A filename or directory name containing an ASCII 255 will foil casual snoopers, because anyone who attempts to list a file or shift to a directory to which you've added an "invisible" filename character will receive the DOS error message "Bad command or filename."

The same technique can help safeguard a new DOS directory. For instance, if you want to protect a directory named HELLO, type:

```
C:>MD HELLO
```

then hold down Alt and, from the numeric keypad, type the number 255. When you release the Alt key, DOS adds a blank character after the directory name.

Entering ASCII characters from the keypad also is handy for typing graphics characters (those that draw interconnected lines and boxes) into a text file. Almost all DOS manuals provide a list of keyboard and graphics characters and their corresponding ASCII values.

—John Wolfskill

DISK ROAD MAPS

Debug lets you read and write disk information piece by piece, bypassing DOS's familiar file system. Using the instructions in the main article, you can read from and write to disk by sector. (Sectors are 512-byte lengths of a disk's concentric tracks; they sometimes are grouped into clusters.) The table shows the basic information you need to find your way through the sector layouts of the four kinds of floppy disks. Hard drive specs vary too widely to be included here, and playing with them can lead to total data loss. If you absolutely must use Debug's sharpest tools on your hard disk, you'll generally find sector specifications in the computer manual.

FLOPPY DISK LAYOUTS

	5¼-Inch		**3½-Inch**	
Disk Capacity	**360K**	**1.2MB**	**720K**	**1.44MB**
Bytes per sector	512	512	512	512
Sectors per cluster	2	1	2	1
Bytes per cluster	1,024	512	1,024	512
Total number of clusters	354	2,371	713	2,847
Root directory's starting sector	5	15	7	19
First data sector	12	29	14	33

ory, where Debug lets you work on it. From there, you must write the changes back to disk when you're done. That's the scary part.

If you type the following command to display the first sector of your floppy disk, now in memory:

```
D100 L200
```

Debug displays 200 hex (512) bytes, or one sector of your disk. (Standard 3½- and 5¼- disks have 512 bytes per sector, but other elements of sector layouts vary (see the sidebar, "Disk Road Maps").

Look toward the bottom of the screen, and you'll see the classic error message displayed when you try to boot from a nonbooting disk: "Non-System disk or disk error..." You're going to use the E command to change that message in memory, then write the new data out to disk. Type the following single line of code:

```
E2A0 "Not a boot disk, sucker!"
  0D 0A "Press any key" 00
```

This pokes a put-down into memory locations starting at 2A0 hex. The 0D and 0A put a carriage return/line feed between the two quoted strings of text. (The last byte must be zero, as shown above.) If you create your own message, note that you have only about 64 characters available; enter more text than that and you'll overwrite some important information on disk. The line above is about 40 characters long.

Once you poke in the new text, you can view it again using D100 L200. Confirm that it hasn't gone beyond offset 2E5. If it has, quit Debug and start over—nothing will be lost. If everything is OK, use the following command to write the data back to disk:

```
W100 0 0 1
```

This is essentially the same command used to load the sector into memory; typing *W* instead of *L* writes one sector back to drive A. After that's done, quit Debug and return to DOS. Reset your computer with the disk still in drive A and be prepared to be insulted.

Don't Touch That Dial

In the next issue, I'll wrap up this Debug discussion with some old-fashioned programming. I'll break the code on those cryptic Debug scripts you find listed in magazines, and I'll include a few scripts of my own to make your computing life much easier.

Dan Gookin is a free-lance author who lives in Issaquah, Wash. His upcoming book, Dan Gookin's Book of Underground DOS, *from Bantam, goes into great depth on Debug and all sorts of cool stuff about your computer.*

TIP SPEEDY DIRECTORY SEARCHES

Do you want to make DOS more responsive when searching for files located in your directory path? The next time you need to create more than a single directory to hold a group of new files, create all of the subdirectories first, before you copy files to them. This lets DOS place the subdirectory filenames together at the beginning of the disk's data platter and closer to the file allocation table (FAT). In most cases, you'll find that DOS then will spend less time exercising your hard drive as it searches through hundreds of files to reach the subdirectory where your file resides.

—J.W.

Advanced Class In Copying Files

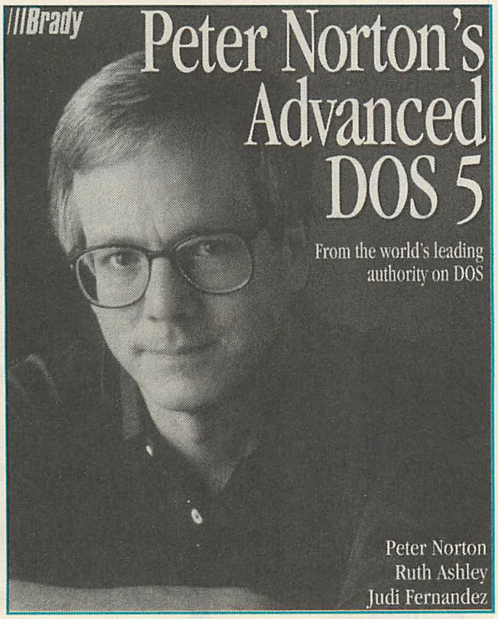
Sit down and pull out your highlighting pen. This discussion of XCOPY, COPY, and REPLACE is worth taking notes on.

by Peter Norton, Ruth Ashley, Judi Fernandez

DOS's copying commands have evolved over the years. Joining COPY, which has many limitations, are the much more sophisticated XCOPY and REPLACE commands. If your knowledge of file duplication techniques hasn't kept pace with changes in DOS, perhaps you should spend a few moments bringing yourself up to date. (See the table "Comparing Copying Commands," on page 28, for a summary of the advantages and disadvantages of DOS's copying commands.)

COPY and XCOPY are the two commands that do garden-variety copies, and XCOPY is the command of choice for a number of reasons:

- When copying multiple files, COPY reads then writes one file at a time, while XCOPY reads as much data as memory will hold, then writes that much data, making this a faster method for duplicating files.
- COPY can select files only by name, but XCOPY also can select files by their archive attribute and time/date stamp. This feature lets you use XCOPY to back up files on your hard disk.
- COPY can select files only from one directory, while XCOPY can copy an entire branch, creating subdirectories under the target directory as needed to duplicate the source branch.
- XCOPY will create the target directory if necessary.
- Sometimes empty files are created by an application. An empty file has a directory entry, including a name, but contains no data. COPY won't copy an empty file, but XCOPY will. Which is better? That depends on what you want.
- Unlike COPY, XCOPY sets exit codes. Such codes accompany many DOS commands and indicate whether an operation was successful—an advantage when you're copying files in batch programs, and you plan to erase the source files after copying them.



But COPY can do things that XCOPY can't, namely concatenate (combine) files, change the time/date stamp for a file, and copy to and from nondisk devices.

REPLACE is used only in special situations. It doesn't do everyday copying, but it will seek out and replace with a newer version all the hard disk copies of a file. This function is used primarily to upgrade software that might appear several times on your hard disk. REPLACE's other function compares two directories and copies missing files from

Editor's Note: *This discussion of duplicating files is excerpted from* Peter Norton's Advanced DOS 5, *by Peter Norton, Ruth Ashley, and Judi Fernandez. Copyright 1992 by Prentice Hall Computer Publishing. Published by Simon & Schuster Inc. Reprinted with permission; all rights reserved.*

the source to the target disk or directory. This function can replace files that you accidentally deleted.

As you can see, XCOPY holds an advantage, unless you need one of the special functions offered by COPY or REPLACE.

XCOPY Vs. DISKCOPY

It's often desirable to copy an entire disk. The DISKCOPY command makes a track-by-track copy of one disk on another disk of exactly the same size. XCOPY also can be used to copy entire disks on a file-by-file basis, with these advantages:

- You don't need to use the same size source and target disks. You can copy a 1.2MB disk to a 1.44MB disk, for example.
- XCOPY doesn't copy the source disk's formatting, which might not be accurate for the target disk. For example, DISKCOPY copies bad sector markings from the source to the target disk, blocking out sectors on the target disk that probably aren't bad.
- DISKCOPY fails if the target disk contains bad sectors; XCOPY doesn't.
- With DISKCOPY, if files are fragmented (split into separate parts) on the source disk, they'll have the same fragmentation on the target. XCOPY actually defragments files as it copies them, as long as the target disk was empty to start with.

THE MANY USES OF XCOPY

The table on page 31 shows the format of the XCOPY command (which some people pronounce *excopy,* some pronounce *cross-copy*, and some pronounce *extended copy*). In its simplest form, it copies a file or set of files from one place to another. The following command copies all the DOC files from the current directory to the default directory on drive A, where they will have the same names they have in the current directory:

```
XCOPY *.DOC A:
```

XCOPY will respond with a message like the following:

```
Reading source file(s)...
VOLUMES.DOC
ENCOUNTR.DOC
  2 File(s) copied
```

Copying Branches

One of XCOPY's strengths is copying complete branches of a subdirectory structure. Suppose you want to copy a branch from drive C to drive D. You don't even have to create the target directory first; XCOPY will do it for you. After turning off all hidden and system attributes (because XCOPY won't copy hidden or system files), use a command such as:

```
XCOPY C:\JNFSET D:\JNFSET /S /E
```

If the D:\JNFSET directory does not exist, XCOPY starts by asking this question:

```
Does D:\JNFSET specify a file name
or directory name on the target
(F = file, D = directory)?
```

In this case, you need to type a *D* for *directory*. XCOPY creates the directory and copies all files and subdirectories to D:\JNFSET so that it ends up being a duplicate of the JNFSET branch on drive C.

If D:\JNFSET already exists, XCOPY uses the existing branch, creating subdirectories only if they're missing from the target branch.

If you include a file specification with XCOPY and /S, only the specified files are copied to the destination. If you omit /E, directories that won't receive any files or subdirectories aren't created. With /E, all target subdirectories are created, whether or not they are empty.

Note: One way to create a new directory is to copy files to it with XCOPY. If the specified directory doesn't exist, XCOPY creates it.

Copying a File to the Same Directory

XCOPY can copy a single file to the same directory as long as you assign it a new name. In this case, XCOPY always asks the "file or directory" question, to which you answer *F* for *file*. (If you answer *D*, XCOPY creates a new directory and copies the file to the new directory using its old filename.)

You can avoid the file-or-directory question, however, if you put a backslash after the destination path so that it can't be mistaken for a filename:

```
XCOPY C:\JNFSET D:\JNFSET\ /S /E
```

Avoiding "Disk Full" Termination

One disadvantage that COPY and XCOPY share is the irksome habit of terminating themselves when the target disk is full. Suppose you indicate that all DOC files be copied to drive A, and COPY or XCOPY copies half of them before the disk fills. Then the command quits, returning you to the DOS prompt. If you insert another

COMPARING COPYING COMMANDS

Command	Advantages	Disadvantages
COPY	Can concatenate files Can change the time/date stamp of a file Can copy to and from nondisk devices	Slower than XCOPY Selects files only by name Handles only one directory at a time. Doesn't set an exit code Terminates when the target disk is full Won't distribute a large file over two or more disks Has no prompt facility Will replace files of the same name on a target disk without warning
XCOPY	Faster than COPY Will copy a disk's branch structure (including empty subdirectories, if requested) Selects files by name, date, and/or archive attribute Will turn off the archive attributes of copied files, if requested Sets an exit code Will prompt you with the names of files selected for copying	Terminates when the target disk is full Can't concatenate files Can't change the time/date stamp Can't copy to or from nondisk devices Won't distribute a large file over two or more disks Will replace files of the same name on a target disk without warning Sometimes has to ask whether the target is a file or a directory
REPLACE	Finds and replaces target files having the same names as the source files Can limit replacements to newer versions of files Will identify missing files on the target disk or directory and copy them from the source Will prompt you before adding or replacing a file, if requested Will replace read-only files Sets an exit code	Won't replace hidden or system files

disk in drive A and reenter the same COPY or XCOPY command, the same files are copied a second time, because the program doesn't know that some of the files already were copied. Your only recourse is to figure out which files weren't copied and copy them one by one.

You can avoid this problem in several ways. One solution is to use the DOS shell, which lets you continue a copy operation after changing the target disk. Another solution uses BACKUP, but that probably presents more problems than it's worth. A third solution uses the archive attributes with XCOPY. The drawback to this method is that it alters the archive attributes of the source files, which can affect your system for backing up files later on.

The XCOPY procedure has three steps:

- Turn on the archive attribute of every file you want to copy. (Also, turn off hidden and system attributes if you want to copy those files.)
- Start XCOPY for those files using the /M switch to turn off the archive attribute of each successfully copied file.
- When the current disk is full, insert a new disk and repeat step 2. The files that were copied will not be recopied, because their archive attributes are now off.

A batch file to automate this procedure might look like the one in the Program Listing. (%1 is the filespec of the files to be copied.) This procedure prints out the source files' original attributes so you can restore them later, if necessary.

You could make this job more useful by adding two features to the batch file: the ability to accept the target's drive name or pathname as %2 and the ability to accept *Y* or *N* as %3 to indicate whether or not hidden and system files should be copied.

Pruning and Grafting

The tree surgeon's terms, pruning and grafting, are used in PCs to describe the movement of a directory or branch from one parent to another. If you don't have a third-party utility to prune and graft, you can do it using XCOPY, although the process is time-consuming:

- Document hidden, system, and read-only attributes so they can be restored later, if necessary:

```
ATTRIB path1\*.* /S > PRN:
```

(*Path1* is the directory or branch to be moved.)

- Remove these attributes:

```
ATTRIB path1\*.* /S -H -S -R
```

- Copy the entire branch to the new parent directory (which XCOPY will create if it doesn't already exist):

```
XCOPY path1 path2 /S /E /V
```

(*Path2* identifies the new parent.)

- After making sure the branch was copied correctly, remove each directory in the old branch. Start with the lowest directories in the branch and work upward. To delete all the files in a directory, use a command like the following:

```
ECHO Y | DEL path\*.*
```

Then, remove the directory:

```
RD path
```

- Using the printout from step 1, restore the hidden, system, and read-only attributes to each file as necessary, using:

```
ATTRIB filespec attribute(s)
```

It's possible to create a batch file to handle steps 1 through 3, but the last two steps can't be done in a batch program, because you won't know the correct paths, file specifications, or attributes in advance. You could create a batch program to do a simple prune and graft, where the directory has no children and no hidden, system, or read-only files. But more complex jobs must be done manually.

Although, you can rename directories from the command prompt using XCOPY, in the final analysis you can do this much more easily from the DOS shell or with a third-party utility

Verification

Mistakes can occur during a copy operation as a result of hardware failure, disk media failure, or power fluctuations. Mistakes are rare as long as your drive is healthy, but they do happen. By default, DOS doesn't verify copies. The /V switch tells XCOPY to verify the copy, which slows down the duplication process. You should be aware that this is not 100-percent effective. It will catch most, but not all, errors.

Clobbering Files With XCOPY and COPY

Both COPY and XCOPY have one problem in common: They will overwrite an existing file without warning. For example, if you copy A:\MEMBERS.DB to C:\SFGROUP\MEMBERS.DB, but SFGROUP already contains a file named MEMBERS.DB, both COPY and XCOPY replace the existing file with the copy, and you don't receive a message saying the replacement was made. You can't rescue the clobbered file using DOS 5.0's UNDELETE command, either.

You might think you can write a batch program using IF EXIST to find out if a file with the same name already exists in the destination directory, but a total solution is not possible because of the variety of ways in which the source and destination can be expressed in an XCOPY command. The routine below works if the source (%1) is a single filename or a global filespec and no pathname is specified, if the target (%2) is the pathname of another directory, and if no XCOPY switches (such as /S) are desired.

Program Listing. This batch file turns off the archive attributes of files as you copy them.

```
@ECHO OFF
REM Record the current attributes in case
REM you want to restore them when the job
REM is done.
ECHO Current attributes for the %1 files: > PRN
ECHO. > PRN
ATTRIB %1 /S > PRN
REM Turn on the archive attributes for all %1
REM files. Also, turn off their hidden and
REM system attributes.
ATTRIB %1 /S +A -H -S
:COPYLOOP
REM Copy all the %1 files that have positive
REM archive attributes. Turn off attributes
REM as you copy:
ECHO Insert the next target disk in drive A:
PAUSE
XCOPY %1 A: /S /M

REM Repeat loop if XCOPY terminates unsuccess-
REM fully for any reason.
IF ERRORLEVEL 1 GOTO COPYLOOP

:ENDING
ECHO**********************************************
ECHO*                                            *
ECHO*  The archive, hidden, and system attrib-   *
ECHO*  utes were removed from the source files.  *
ECHO*  Check the printout and restore attributes *
ECHO*  as needed.                                *
ECHO*                                            *
ECHO**********************************************
```

End

```
REM Warn user about files that won't be copied:
FOR %%F IN (%1) DO IF EXIST %2\%%F
ECHO %%F won't be copied because a target version
ECHO already exists

REM Copy files that won't clobber existing copies:
FOR %%F IN (%1) DO IF NOT EXIST %2\%%F
XCOPY %%F %2
```

Burying the XCOPY command in the FOR command slows down the copying procedure tremendously, but it's necessary to check one file at a time with an IF [NOT] EXIST statement; otherwise the results could end up being wrong.

Using FOR, IF, and CALL, you can create a batch program that gives the user the option of copying a file when a target version exists, but that makes a complex job out of a capability that's already available with the REPLACE command, as you'll see at the end of this article.

Moving Files

Moving files involves copying them, then deleting the source files. You can write a macro or batch file program to do this easily, as long as you're prepared to assume that the copies were completed successfully. Most of the time, they will be, but as you know, they very well might not be.

The following batch program (a macro can't do the same thing) uses IF ERRORLEVEL to make sure all files are copied successfully before the source files are deleted:

```
@ECHO OFF
XCOPY %1 %2
IF ERRORLEVEL 1 GOTO NOMOVE
IF "%2"=="" GOTO NOMOVE
DEL %1
GOTO ENDING

:NOMOVE
ECHO***************************************
ECHO An error occurred during the copy
ECHO operation, so the source files were
ECHO not deleted.
ECHO***************************************

:ENDING
```

The second IF command makes sure that a destination is specified. If the source is in the current directory and no destination is specified, XCOPY terminates itself with a "File cannot be copied onto itself" message and, unfortunately, issues an exit code of zero. You don't want the source files to be deleted in this case, but the exit code won't prevent it. So you have to weed out this case separately. Unfortunately, it also weeds out situations in which the source is not the current directory and it is safe to omit the destination directory. But there is no way to identify that situation, and it's better to be safe than sorry.

This program doesn't permit you to include switches in the original command. You can revise the program to allow switches such as /A or /D, but you then must weed out inappropriate switches such as /P (because with this switch, source files would be deleted whether or not they are copied) and /S (because DEL can access only one directory at a time).

THE UNIQUE COPY COMMAND

The COPY command has some features you can't find anywhere else in DOS:

- It can concatenate files; that is, it combines two or more files into one.
- It can update the time/date stamp on a file.
- It can copy to and from nondisk devices.

Because XCOPY is more useful for day-to-day copying tasks, this discussion covers only COPY's unique features (see the table on page 33 for a summary of these capabilities).

Concatenating Files

Suppose you want to combine three ASCII files named PART1, PART2, and PART3 into a new file named REPORT. The following command will do it:

```
COPY PART1 + PART2 + PART3   REPORT
```

Copy displays the following information as it works:

```
PART1
PART2
PART3
1 File(s) copied

C:\DOCS>
```

The message says only one file was copied, because only one file was created (or replaced). The following command may or may not have the same effect:

```
COPY PART? REPORT
```

Since a global source filespec is used with a single target filespec, the plus sign is not necessary. COPY will concatenate all matching source files into the one target file. This might include more files than you intended, however.

Also, the order in which the files are concantenated may differ. That's because, when creating a new file, DOS places it in the first available directory entry, which might be the former entry of a deleted file. For this reason, the order of entries in a busy directory seems arbitrary. Even if you write PART1 first, PART2 next, and PART3 last, these three files could appear in any order in the directory and thus appear out of order in your new file.

Appending Files

Sometimes you want to concatenate files by appending them to an existing file. For example, suppose you want to append PART2 and PART3 to PART1 instead of creating a new file. You can omit the target filespec from the command, as in:

```
COPY PART1 + PART2 + PART3
```

If COPY can't find PART1, it appends PART3 to PART2 (if PART2 exists). When a target filename is not given, COPY uses the name of the first file mentioned in the command. If it can't find that file, it uses the name of the first file it finds.

The following command is legitimate but might not have the same result as the previous one, because a different set of files might be identified, and they might appear in a different order. The target file will be the first file in the directory that matches the PART? filespec:

```
COPY PART?
```

But you can control the target filename by identifying it:

```
COPY PART? PART1
```

This command could create a problem, however. Because PART1 is the target filename, COPY begins appending each source file to it. After the first file is appended, PART1 is changed. If COPY then finds the PART1 entry in the directory and realizes it matches the filespec for a source file, it knows that it can't append PART1 to itself. The following message is displayed:

```
Content of destination lost
  before copy
```

This message looks devastating, but it's normal. It means that, because the target file already was changed, it cannot be used as a source file. In fact, things are progressing just as they should, and you can ignore this message.

The Effect of Concatenation On Certain File Types

If PART1, PART2, and PART3 are ASCII (unformatted) text files,

XCOPY

Copies files and subdirectories.

Format:

XCOPY *source* [*destination*] [*switches*]

Parameters and Switches:

source	Identifies the files to be copied.
destination	Identifies the location where the copy should be stored. If omitted, the current directory is used.
/A	Copies only files with a positive archive attribute. The archive attribute is not affected by the copy procedure.
/M	Copies only files with a positive archive attribute. The source file's archive attribute is turned off after the copy is completed.
/D:*date*	Copies only files with a date stamp on or after the specified date.
/P	Prompts you for permission to copy each selected file.
/S	Copies subdirectories and their files to the target directory.
/E	Copies empty subdirectories. (If this switch is not included, only subdirectories containing at least one source file are copied.) This switch may be used only with /S.
/V	Verifies each copy operation.
/W	Displays a "Press any key to begin copying file(s)" message and waits for the user's response.

Notes:

Source can be a drive name, pathname, global file specification, or individual filespec.

Destination can be a drive name or pathname. If the source is an individual filespec, the destination can be an individual filespec, in which case the new copy receives the specified name.

A file can be copied into the directory in which it resides only if you give the new copy a different filename.

If *destination* identifies a directory that doesn't exist, XCOPY creates the directory and copies the source files to it.

Hidden and system files aren't copied, but empty files are.

Archive attributes are set during a copy procedure.

XCOPY sets an exit code.

/A and /M are mutually exclusive.

they'll concatenate without difficulty. Other types of files are not so easy.

If PART1, PART2, and PART3 were created in a text editor and stored in its native format, they probably have headers and trailers. When you concatenate the files, all three sets of headers and trailers are in the file, creating a hybrid that can't be processed by the original editor. This problem exists with files created by most of today's applications: desktop publishers, spreadsheets, databases, and so on. If you want to concatenate such files, don't use COPY; concatenate the files from within the application so that it can create (and remove) headers and trailers.

As with data files for applications, you don't produce a usable file by combining binary files, such as executable program modules. Concatenating two program modules not only produces an invalid program module, it might create a monster that, if executed, will run amok inside your system. Here again, if you want to combine files, use the proper tools. For example, use a linkage editor to combine two executable program modules.

Updating Time/Date Stamps

When a file is copied using any of the DOS copying commands, the file's time/date stamp is unchanged, the theory being that the time and date identify the file's version. But when you concatenate files, the current time and date are assigned to the newly created file, because it is a new version.

You can trick DOS into assigning the current time and date to a file by pretending to concatenate it with a command in this format:

```
COPY source filename+,, [destination]
```

Be sure to use the name of a single source file, because a global one would cause a real concatenation to take place. (If you want to update the time/date stamps on a group of files, you can use the FOR command to get everything done with one command.) The two commas after the plus sign force DOS to stop concatenating. If you include a destination, the copy is made and only the target file receives the new time/date stamp. If you don't provide a destination, the source file receives the new time/date stamp.

The following batch program changes the time/date stamp to any value you wish on a group of files. %1 is the new time, %2 is the new date, and %3 through %9 are optional filespecs, which can be global. (The COPY command's /B switch is explained shortly.)

```
@ECHO OFF
TIME %1 > NUL
DATE %2 > NUL
FOR %%F IN (%3 %4 %5 %6 %7 %8
  %9) DO COPY /B %%F+,,
ECHO The system time and date
ECHO have been changed.
ECHO Please reset the correct
ECHO time:
TIME
ECHO Please reset the correct
ECHO date:
DATE
```

Matched Concatenations

COPY permits a special technique that will concatenate files with matching filenames but different extensions. The following example should make this clear:

```
COPY *.TXT + *.ADD *.NEW
```

This command causes COPY to concatenate FORM.TXT to FORM.ADD, creating FORM.NEW; combine TEACH.TXT and TEACH.ADD, creating TEACH.NEW; and so on. If the directory contains TEACH.ADD but not TEACH.TXT, or vice ver-

TIP GONE, BUT NOT FORGOTTEN

You may or may not know this, but DOS 5.0's DEL command does not delete files immediately. Instead, it marks them as inactive and frees up the disk space they previously occupied for subsequent use. Normally, this isn't a problem.

When searching directories for a particular file, however, DOS checks inactive, as well as active, entries. This can increase the amount of time it takes to find a specific file.

Here's an easy way to eliminate this potential slowdown. Enter the following command to display a list of a directory's inactive files:

```
UNDELETE *.* /DOS /LIST
```

If several inactive files exist, follow these steps to remove deleted files from a directory:

1. Use the DOS XCOPY command to copy all active files to a temporary directory. (XCOPY ignores inactive files.)
2. Type:

   ```
   DEL *.*
   ```

   to delete all files in the original directory.
3. Using the RD command, remove the original directory.
4. With the MD command, re-create the original directory.
5. Use XCOPY again to copy the initial files from their temporary location to the re-created directory.

—Jack Nimersheim

sa, an error message is displayed and TEACH.NEW is not created.

There are a few variations on matched concatenations. You can, for instance, specify more than two source files, as in:

```
COPY *.1 + *.2 + *.3 + *.4  *.ALL
```

Or you can omit the target filespec to append files to the first file mentioned, using a command such as:

```
COPY *.TXT + *.ADD
```

Copying to and From Devices

COPY lets you copy files to and from devices, using DOS's accepted device names. A quick-and-dirty way to print a file without using DOS's PRINT command (which loads a TSR into memory and formats the printout) is a command such as this one:

```
COPY filespec PRN
```

You also can copy files to and from serial ports, but the result is unlikely to be satisfactory, because a communications program isn't acting as an intermediary.

You can concatenate files while copying to a device. The following command, for example, concatenates all TXT files and prints the result:

```
COPY *.TXT PRN
```

The ABCs of /A and /B

The /A and /B switches can be important in copying and concatenating files, but to use them properly, you have to understand a bit about DOS file storage.

DOS stores files in disk storage blocks called clusters, or allocation units. Every disk has a fixed cluster size, which depends on the size of the disk and the program that formatted it. For example, a small-capacity (360K) floppy disk has 512-byte clusters, but a higher capacity disk, such as a 40MB hard disk, might have 2,048-byte (or larger) clusters. The cluster size is set during formatting.

DOS allocates disk space to files in whole clusters, but a file needn't occupy a complete cluster. The leftover space at the end of the cluster is called slack. On a disk with 2,048-byte clusters, a 1-byte file occupies one whole cluster and has 2,047 bytes of slack. A 15K file will occupy eight clusters with 1K of slack at the end of the last cluster. Slack space is wasted space; it cannot be allocated to any other file. It might contain left-

COPY (for concatenation)

Combines two or more files into one.

Format:

COPY [*switches*] *source* [+...] [*destination*] [*switches*]

Parameters and Switches:

source	Identifies the file(s) to be copied.
destination	Identifies the new file(s) to be created.
/A	Uses ASCII mode for the copy.
/B	Uses binary mode for the copy.

Notes:

For the full COPY format, see your DOS reference manual.

For concatenation to occur, the source must either be a global filespec or multiple filespecs connected by plus signs.

If the source is a global filespec, all files matching the filespec are concatenated.

If the destination is omitted, files are concatenated into the first source file that DOS locates.

The current time and date are assigned to the destination file, which also has a positive archive attribute.

The following special format updates the time/date stamp of an individual file:

```
COPY [/B] filespec+,, [destination]
```

The following special format concatenates files with identical filenames but different extensions.

```
COPY [switches] *.ex1+*.ex2[+...] [*.ex3] [switches]
```

The /A or /B switch affects the filespec preceding it and all subsequent filespecs, until it reaches a filespec that is followed by another /A or /B switch.

When /A is applied to a source file, the file is read up to, but not including, the first EOF marker (1A hexadecimal). When /A is applied to a destination file, an EOF marker is added to the end of the file. When concatenating, COPY assumes by default that /A is the switch for both the source and destination files.

When /B is applied to a source file, the file's size is determined by its directory entry, and any embedded EOF markers are ignored. When /B is applied to a destination file, no EOF marker is added to it.

over data from previous files that occupied more of the cluster.

When DOS reads a file for copying, printing, concatenation, or some other operation, it wants to read only the valid file data and ignore the slack. DOS has two methods of identifying the end of legitimate data in a cluster: ASCII and binary.

The ASCII method depends on an end-of-file (EOF) marker—which is ^Z, or 1A hex—appearing as the last character in the file. When using the ASCII method, DOS scans the data it reads and stops when it encounters 1A hex. (If a file has no EOF marker, DOS uses the file's size to decide when to stop reading.)

The binary method uses the file's size as recorded in its directory entry. If the directory entry says that the file occupies 3,267 bytes, DOS will read exactly 3,267 bytes from the disk. Any 1A hex characters in the data are read just like other data.

Unfortunately, the ASCII method works only with files that do not include inadvertent EOF markers within their data, as might happen with binary files, worksheets, graphics files, and other application files. The only files that work reliably with the ASCII method are true ASCII files, those containing only ASCII character data. They may not contain formatting information.

DOS also has an ASCII and binary method of *writing* files. In the ASCII method, DOS adds the EOF marker to the end of the file. In the binary method, it doesn't.

Some DOS utilities (especially the older and more elementary ones) use the ASCII method, while others use the binary method. Commands, such as TYPE and PRINT, that are designed to work with files in ASCII format use the ASCII method. Those such as XCOPY and BACKUP, which should work with any type of file, use the binary method.

COPY Methods

COPY sometimes uses one method and sometimes the other. It uses the binary method for most copy processes. But for processes that make sense only with ASCII files, such as concatenation and copying to the console (CON) or the printer (PRN) it uses the ASCII method. You can force it to use the alternate method using the /A or /B switches.

You can force COPY to use the ASCII method to read a source file by including the /A switch before or after the source filespec. You can force COPY to use the ASCII method of writing a file, which adds an EOF marker to the end of the copy, by including the /A switch after the target filespec. The EOF marker never is essential to DOS programs, but you might have some other software that requires it. (Most modern programs don't.)

You can force COPY to use the binary method to read a source file by including the /B switch before or after the source filespec. If you have a reason to concatenate binary files, you must use the /B switch to force COPY to read the source files in binary mode, since COPY concatenates in ASCII mode by default.

You can force COPY to write a file in binary mode, which omits the EOF marker, by placing a /B after the target filespec. You must use this switch when concatenating binary files if you don't want DOS to add an EOF marker to the end of the new file.

Date Stamping Non-ASCII Files

You are most likely to need the /B switch when updating the time/date stamp for a non-ASCII file. If you don't use /B, the file might be truncated by the pseudo-concatenation; it also will receive an EOF marker. Specify a /B before the filespec to make sure the whole file is copied with the new time/date, but that no EOF marker is added. The following example updates the time/date stamp on the file named DEFRAG.COM:

```
COPY /B DEFRAG.COM+,,
```

WHEN TO USE REPLACE

REPLACE has two special functions, both of which rely on its ability to select files by comparing the contents of the source and target directories. It can replace files that are present in the source directory but are missing in the target directory. (Here *replace* is used in the sense of restoring something that was lost.) It also can replace files that exist in both the source and target directories. (Here *replace* is used in the sense of substituting one item for another.)

The table at right shows the format of the REPLACE command.

Replacing Missing Files

Suppose you deleted several files from C:\PROPS and now you have changed your mind. It's too late to undelete them, but you have backup copies on a floppy disk. (If the

TIP WINNING THE SHELL GAME

The SHELL statement in a CONFIG.SYS file can do more than increase environment space, probably its most common use. It also lets you load a command processor other than DOS's—The Norton Utilities' NDOS, for example.

When you use a SHELL statement to load a command processor, however, you should make certain it ends with a /P switch. The /P switch makes permanent the specified command processor. Without it, for one thing, DOS may ignore your AUTOEXEC.BAT file during start-up.

—J.N.

backups were made with BACKUP, the following technique won't work.) If the files fit a generic filespec, you can restore them with an XCOPY command such as:

```
XCOPY A:*.DOC C:\PROPS
```

Notice that this command can replace files that you don't want to replace. If a generic filespec or two won't select the desired files, then REPLACE might be the easiest way to restore them. The following command, for instance, selects all files that exist on drive A but do not exist on C:\PROPS:

```
REPLACE A: C:\PROPS /A /P
```

Because the command uses the /P switch, REPLACE displays each filespec and lets you decide whether or not to copy it. This helps avoid inadvertent replacements.

Replacing Existing Files

Suppose you installed a particular application in several places on your hard disk. When you receive an updated version of the application, you can use the following command to substitute the new version for the old versions:

```
REPLACE A: C:\ /R /S
```

This seeks out and replaces all the files on the hard disk that have the same name as the files on the disk in drive A. REPLACE's /U (for *update*) switch causes a file to be replaced only if its time/date stamp indicates that it is older than the matching source file. This comes in handy mostly with certain backup methods.

Using REPLACE With /P To Avoid Clobbering Existing Files

In a continuing effort to create a batch file program that copies files without clobbering files that already exist, let's see if REPLACE with the /P switch will do the trick. The following batch file uses REPLACE twice, the first time to replace existing files with permission; the second time to add missing files (no permission is needed). The overall effect is to copy all files identified by %1 to the destination. (To avoid premature termination of the first command, even read-only files are replaced in the new destination.)

```
@ECHO OFF
REPLACE %1 %2 /P /R
REPLACE %1 %2 /A
```

This batch program can handle only one subdirectory at a time, so it lacks a lot of XCOPY's punch, and it's also extremely slow when compared to XCOPY, but it is a safe way to copy files from one directory to another.

Practice Makes Perfect

That's the end of today's lesson. Now for your homework: incorporating these techniques into your daily computing sessions. After all, here—as elsewhere in life—practice makes perfect sense. ■

REPLACE

Replaces destination files with source files of the same name or copies source files that are missing in the destination directory.

Format:

REPLACE *source* [*destination*] [*switches*]

Parameters and Switches:

source	Identifies the files to be copied to the destination. A filespec must be included.
destination	Identifies the directory to receive the copies. May not include a filespec. If this parameter is omitted, the current directory is assumed.
/A	Adds missing files to the destination directory. Do not use with /S or /U.
/P	Prompts for confirmation before replacing a file.
/R	Permits replacement of read-only files.
/S	Replaces matching files in destination subdirectories, as well. Do not use with /A.
/W	Displays a message and waits for the user's response before beginning replacement.
/U	Examines time/date stamps of matching source and destination files and replaces (updates) only destination files that are older than the source files. Do not use with /A.

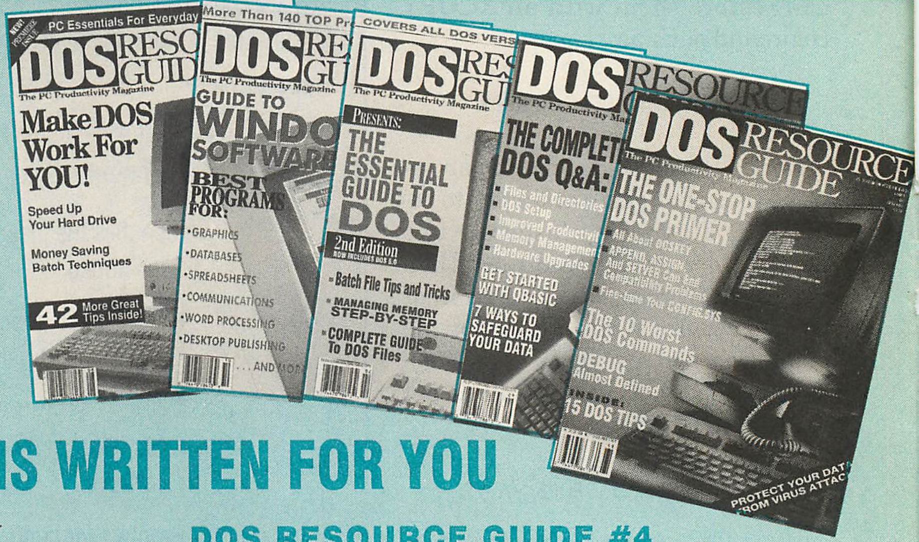
Note:

REPLACE terminates automatically if it is unable to replace a file because of the read-only attribute. Use the /R switch to avoid this type of termination.

Peter Norton has published dozens of best-selling books on computing and is recognized as the industry authority on DOS. Award-winning authors Ruth Ashley and Judi Fernandez have over forty books to their credit.

TAKE THE DREAD OUT OF DOS!

If you're trying to succeed with DOS, but feel like a General Failure...



DOS RESOURCE GUIDE IS WRITTEN FOR YOU

DOS RESOURCE GUIDE, the bi-monthly periodical packed with tips, tutorials, and information on all aspects of PC computing, is your antidote for computing confusion.

DOS RESOURCE GUIDE #1

Make DOS Work for You

Covers six areas of computing that are vital to DOS users: start-up files (AUTOEXEC.BAT and CONFIG.SYS), batch file creation, hard drives, safe backups, organizing file management, and hardware maintenance. Includes: List'em, a batch file that helps you keep appointments; 10 tips to keep your hard drive healthy; speed-up tips for a logy hard drive; a batch file that automates backups; DOS shortcuts, hardware dos and don'ts. Plus 42 provocative pointers on DOS.

DOS RESOURCE GUIDE #2

Guide to Windows Software

A compendium of software briefs that tells you what's up in the world of Windows. (Even a DOS user has to keep up with the competition.) Guide #2 briefly reviews over 140 programs for Microsoft Windows, including word processors, communications programs, financial and mathematics programs, management programs, developer's tools, and graphics software. Complete list of manufacturers.

SOLD OUT

DOS RESOURCE GUIDE #3

The Essential Guide to DOS

Who are we to argue with success! Our second edition of this classic is updated to include information on DOS 5. Guide #3 includes an annotated list of DOS files and when to use them; articles that will change your view of CHKDSK, COPY, and XCOPY; a cluster of batch file assists; and some clear-headed advice on organizing files on your hard drive. Plus 14 titillating tips on DOS.

DOS RESOURCE GUIDE #4

The Complete DOS Q&A

Breaks DOS into functional sections: Fundamentals, Set-up, Improving Productivity, Memory Management, Hardware Upgrades, Batch Files, and Using Windows. Each section is followed by a question and answer session on the topic. Plus, seven short programs to protect your data; do-it-yourself memory, advice on CONFIG.SYS and AUTOEXEC.BAT, and beginner's Basic. Not to mention nine tips for the DOS timid.

DOS RESOURCE GUIDE #5

The One-Stop DOS Primer

All you can eat: DOSKEY, SETVER, APPEND and ASSIGN, plus a tutorial on using the ATTRIB command. Part 1 of a Debug series. Our "10 Worst" list of DOS commands. Your introduction to serial ports. Includes a Qbasic Printer Control Program: Just type and run. Sharpen your DOS finesse with 15 type-in tips.

Back issues of DOS Resource Guide are $5.95* postpaid!

Use our 800 number and have your credit card ready!

1-800-238-6580

Or send your check or money order to:

DOS Resource Guide
P.O. Box 3162
Salisbury, MD 21802

*Maryland residents add 5% sales tax. Back issues to Canada and Mexico cost $6.95 (Postpaid). Other International (Surface) $8.00 (Air) $13.00. Checks and money orders must be payable in U.S. funds, drawn on a U.S. bank.

VISA, Mastercard, or American Express also accepted. Please include your card number, its expiration date, and your signature on your request.

Please indicate which issue(s) of DOS Resource Guide you are ordering.

DON'T MISS AN ISSUE!

Programming In DOS's Batch Language

If you haven't started writing your own batch files, read this first article in a series that shows you how to automate life at the DOS prompt.

by Hardin Brothers

You've seen them in magazines and, maybe, picked up a few at a user's group meeting. Each of you has probably used at least one, and some of you have come to rely on them every time you sit down at your computer. I'm talking about batch files—those clever little programs that automate dozens of DOS duties and, generally, make your computing life a lot easier.

If, for some reason, you haven't started writing your own batch files, you haven't begun to tap your computer's true potential. In this series of articles, I'll show you how easy it is to create simple batch files. Once you've learned the batch language and a few concepts about controlling it, you'll need only a little imagination to start creating batch files to hand out at your next user's group meeting.

The Fundamental Process

Whenever you type a command, a DOS program called COMMAND.COM collects your input. If you type one of DOS's internal commands—operations built into COMMAND.COM—COMMAND.COM executes it without delay. If the program you name is not internal to DOS, COMMAND.COM looks in the current directory for a COM, EXE, or BAT file (in that order) with a matching name. It then goes on to check in every directory in your PATH setting, if it doesn't locate a match in the current directory. If it finds a COM or EXE file to execute, COMMAND.COM tells DOS to load and run the program. When the program is finished, COMMAND.COM waits for its next instruction. If COMMAND.COM finds a BAT file to execute, it goes into its batch mode.

When COMMAND.COM is in batch mode, it takes its input from a batch file instead of from the keyboard. Each line of a batch file is simply a command that you could type at the DOS prompt (with a few exceptions, which I'll discuss below). A batch file can take advantage of eight commands (see the accompanying sidebar, "Batch File Commands"), which make up DOS's batch language, and four other COMMAND.COM features. (These are the @ sign and the period after ECHO, both discussed below, and the percent sign with replaceable parameters and environmental parameters, both of which I'll cover in the next issue.)

When COMMAND.COM opens a batch file, it reads one line, closes the batch file, and then executes the line. If the line contains an internal command, that command is carried out; if it contains the name of a program, COMMAND.COM tells DOS to run that program. When the internal command or program is finished, COMMAND.COM opens the batch file, reads the next line, executes it, and so on.

Batch File Prerequisites

For COMMAND.COM to accept a program as a batch file, it must meet three requirements: Its filename must end with the BAT extension, and it must be in ASCII format and divided into lines that contain 127 or

BATCH FILE COMMANDS

DOS's list of batch file commands includes only internal commands, which means you do not have to provide a pathname when invoking them. The numbers in parentheses indicate the version of DOS in which that command first appeared.

CALL (3.3)—calls one batch file from within another.

ECHO (2.0)—turns DOS messages on and off and displays a message on the screen.

FOR (2.0)—loops DOS through a series of files and performs a command once per file.

GOTO (2.0)—transfers program control from the batch file line just executed to the line that contains the name that follows the GOTO command.

IF (2.0)—allows conditional processing of DOS commands.

PAUSE (1.0)—suspends the processing of a batch file and waits for the user to strike a key to continue.

REM (1.0)—lets you include comments in a batch file.

SHIFT (2.0)—assigns a new variable to all replaceable parameters.

—H.B.

fewer characters. You can create or edit a batch file with DOS 5.0's Edit utility, with the Edlin program included with all versions of DOS, or with almost any word processor. If you use a word processor, make sure you save the file in ASCII format, sometimes called DOS format or text format.

Be careful when selecting the name for your batch file. If you give a batch file the same base name as an internal command—by naming it DIR.BAT, for instance—the batch file will never run, because COMMAND.COM looks through its list of internal commands *before* looking for a file to run. And if you give a batch file the same name as a COM or EXE program, that COM or EXE program will run instead (unless the batch file is in a directory that comes earlier in your PATH statement).

TIP STARS BE GONE

Many DOS users know that wildcard characters are useful for listing, copying, or deleting groups of files that contain common filenames or extensions. For example, DOS manuals provide this "official" wildcard syntax:

```
COPY *.* A:
```

to allow you to copy a group of files from the current drive to drive A. If you use DOS 2.11 or later, however, you can use this "unofficial" version of the command:

```
COPY . A:
```

Not only will you impress your fellow DOS aficionados, you'll also save a pair of keystrokes in the bargain.

—J.W.

The Simplest Batch Files

The easiest batch files to write and understand are those that automate a simple and common computer task. For example, my laptop computer extends its battery life by leaving off the power of the internal modem. When I want to use the modem, I have to turn it on, run a communications program, and then turn off the modem's power. It was more practical for me to automate this process so that I'd never accidentally run the communications program when the modem wasn't turned on.

So, I wrote a batch file that I call COMM.BAT. Its three lines (which run a hypothetical communications program called COMMUN.EXE) look like this:

```
MPOWER -ON
COMMUN
MPOWER -OFF
```

You might want to write a batch file that automates a tricky or hard-to-remember command. For instance, you may be one of the many computerists who uses the DOS utility XCOPY to move new and altered files to a floppy disk for storage or backup. XCOPY has a number of switches, however, that can make it difficult to remember the exact command you want. The following one-line batch file, called SAVE.BAT, can help:

```
XCOPY *.* A: /M /P /S /V
```

REM

The easiest batch file command is REM, which is an abbreviation for *remark*. You will find remarks employed in virtually all programming languages to help explain what is happening there and why. In a batch file, you might use REM to claim credit for authorship or to explain how a complex part of a batch file works. If, in the future, you want to change the batch file, the remarks will help you understand what is going on and how to make the necessary changes.

You also can add REM and a space to the beginning of any batch file line that you want to disable temporarily. It is much easier to add REM statements to a series of lines and remove the REMs afterward than to erase a block of lines from a batch file and later try to remember and retype those same lines. (You also can disable CONFIG.SYS lines by placing a REM statement at the beginning of a line.)

PAUSE

Another simple batch file command is PAUSE, which always shows up on a line by itself. When COMMAND.COM executes PAUSE, the message "Press any key to continue" is displayed, and the computer waits for your action. If you press Ctrl-C or Ctrl-Break, COMMAND.COM displays the message "Terminate batch job (Y/N)?" and gives you a chance to exit from the batch file. If you press any other key, the batch file continues with the next line.

You can use PAUSE at any point where you want to provide an opportunity for the user to press Ctrl-C to abort the program. But PAUSE is used most often at times when you might need to change disks, turn on the printer, or perform other simple maintenance functions.

ECHO

The batch file command ECHO has several different uses. If you type the following command from the DOS prompt:

```
ECHO OFF
```

the prompt disappears, although the computer still displays what you are typing and reacts normally to your commands. You can reinstall the DOS prompt by typing:

```
ECHO ON
```

In this respect, it's difficult to think of why you would need ECHO, unless you want your computer to have a blank screen to make it appear as if it's busy or turned off.

In a batch file, however, ECHO has several important uses. Batch files always begin and end in the same echoing state as the DOS prompt. Because echoing normally is turned on at the DOS prompt, it also is on at the beginning of every batch file; this means that the DOS prompt and a line from the batch file are displayed as each line is executed. If you use very simple batch files, you may not care, but you probably don't want to watch the lines of a long and complex batch file scroll across your screen.

Most batch files therefore begin by turning off echoing. To avoid having a batch file's opening ECHO OFF command appear on the screen, you can use (with DOS 3.3 or later) the @ sign, as in:

```
@ECHO OFF
```

This tells COMMAND.COM to execute the current line without echoing it. (As a matter of fact, you can use an @ sign as the first character in any batch file line to execute the line "silently" when echoing is on—just don't place any spaces between it and the next word.)

Another use of the ECHO command is for displaying text on the screen. If a line starts with ECHO, followed by a space and any text other than the words *on* or *off*, all of the text is displayed.

For example, many batch files include these two lines:

```
ECHO Press Ctrl-C to exit or
PAUSE
```

TIP TIME AND DATE BYPASS

It's a good idea to place the DOS TIME and DATE commands in your AUTOEXEC.BAT. Doing so lets you check the time and date and thus verify that files you create or modify will contain a correct time/date stamp. You may not want to reset the time and date whenever your computer boots, however. In addition, you ordinarily must remove the TIME and DATE commands from your AUTOEXEC.BAT if you need to start your computer when no one is present at the keyboard.

To solve this problem, you can use the DOS input redirection symbol (<) to create a small data file that lets you view the output of the TIME and DATE commands, while allowing the start-up process to continue uninterrupted. You simply create an input file, then use DOS input redirection to "feed" the carriage-return character that the TIME and DATE commands require to the keyboard buffer at the appropriate moment.

To create an input file called CR.DAT, for instance, type:

```
COPY CON CR.DAT
```

and press Enter. Next, type Ctrl-M and press F6. Press Enter to write the data file to disk. This produces a small file containing a single carriage-return character. You then need to ensure that your AUTOEXEC.BAT file contains the following lines:

```
TIME < CR.DAT
DATE < CR.DAT
```

(Use a full pathname if you store CR.DAT in a directory other than the root directory.)

Press Ctrl-Alt-Del to reboot your computer and have these commands take effect. The output of the TIME and DATE commands will appear on screen as your computer starts up, but DOS won't ask you to enter new values for the current time and date.

—J.W.

which create the following screen display:

```
Press Ctrl-C to exit or
Press any key to continue
```

The computer then waits for you to press a key.

As you may know, in DOS 5.0, ECHO serves another important function. You can follow it with a period to display a blank line on screen. If you're using an earlier version of DOS, the easiest way to get a blank line is to follow the ECHO command with a space and then type the special character 255, which is displayed as a blank. (To make this special character, hold down Alt and type 255 on the numeric keypad.)

ECHO works with DOS's redirection symbols to send text to a file or to a device such as a printer. For instance, to send text to a printer from a batch file, you can use the following line:

```
ECHO This goes to the printer. > PRN
```

You also can perform the same function from the DOS prompt.

CALL

You may encounter times when you want to use one batch file to start another. The easiest way to do this is to use the name of the second batch file as a command inside the first. For example, FIRST.BAT might contain the line:

```
SECOND
```

which will start your second file, SECOND.BAT. When you use this method, COMMAND.COM forgets all about the first batch file and starts the second file as if the first one never existed.

At other times, you may want to execute all the commands inside the second file and then return to the first batch file and continue. Beginning with DOS 3.3, you can use the CALL command to start the second batch file, as in:

```
CALL SECOND
```

If you're using an earlier version of DOS, you can do the same thing—albeit more slowly—by starting a second copy of COMMAND.COM and letting it run your second batch file. The command to accomplish this would read:

```
COMMAND /C SECOND
```

The /C tells the second copy of COMMAND.COM that it should run a single program or command—in this case, SECOND.BAT—and then exit.

You'll find CALL most useful if you have a group of commands that you want to execute from several different batch files—for instance, if you are using Windows in enhanced mode and want to install a group of memory-resident utilities (TSRs) each time you start a DOS-based program. To do this, you can employ a batch file to start each program. You would begin each of these batch files this way:

```
@ECHO OFF
CALL LOADTSR
```

Now if you want to add or delete a TSR from your normal setup for running DOS under Windows, you need only change LOADTSR.BAT instead of altering every batch file that can be started from a Windows icon.

GOTO It

In the next issues, I will explore the other batch file commands—IF, GOTO, FOR, and SHIFT. I'll also look at replaceable parameters and environmental variables and show you how to enhance your batch files with programs written with Debug.

Right now, you have enough information to create hundreds of handy batch files. Try experimenting with the above commands, altering your AUTOEXEC.BAT file, for example, so that it prints a friendly greeting each time you turn on your computer. Once you start writing batch files, you'll be pleased to discover that your computer can do a lot of work by itself. ■

Free-lance writer Hardin Brothers has been working with and writing about computers for 13 years.

How Basic Can You Get?

If you have DOS, you have a version of Basic, the most popular programming language in the world. And, as an enthusiast who likes to get down to fundamentals, you've probably written a program or two—be it a quickie dashed off to solve a printing problem or a menu-driven home finance program that took weeks to perfect.

We'd like to see the short Basic programs you're most proud of. They must be 100 or fewer lines and written (or rewritten) in QBasic, the Quick Basic interpreter that comes with DOS 5.0. Include a brief explanation of how each program works. We'll print the best ones in a new column in *DOS Resource Guide,* along with a critique by a professional Quick Basic programmer. We're partial to short listings that solve a computing problem or overcome DOS's limitations, rather than application software.

Send your QBasic programs, each with a brief explanation of how it works, to:

DOS Resource Guide
80 Elm St.
Peterborough, NH 03458

Or upload the files to our on-line addresses:

CompuServe (75300,2361)
MCI Mail (469-7260)

Command Performance

Reprogramming your keyboard's function keys is one sure way to get more work done with less effort.

by Antonio and Alfonso De Pasquale

If you feel that you're doing too much typing at the DOS prompt, perhaps you are. Wouldn't it be nice to display a directory by pressing a single key? Or how about invoking any one of a dozen applications by pressing another key? You can do that and more by taking advantage of a few old friends: your keyboard's function keys.

Over the years, these keys have saved the sanity of users who find themselves doing the same tasks—in WordPerfect, in Lotus 1-2-3, in dBase—day in and day out. Unfortunately, DOS doesn't make as good use of your keyboard's function keys as such programs do. Pressing F1 lets you recall the previous command character by character, pressing F2 followed by a backslash (\) recalls the first portion of the previous command, and pressing F3 lets you recall the command in its entirety. But that's really it; the remaining nine keys sit idle.

ANSI.SYS: A program that gives you greater control over your screen display and keyboard by letting you do such things as change foreground and background colors and change key assignments.
Program Type: Device driver, external to DOS.
Available: MS-DOS 2.0 and later.

CONFIG.SYS: A text file containing system setup information that DOS uses each time you boot up your computer.
Program Type: Auxiliary text file.
Available: MS-DOS 2.0 and later.

With a little help from DOS's ANSI.SYS program, you can put these keys to work and pair them up with an Alt, Ctrl, or Shift key if you need more single-keystroke commands at your disposal. The techniques we describe require neither expert knowledge of DOS nor high-powered equipment. We've programmed function keys on systems as varied as 8088 machines with 512K of RAM and multimegabyte 386 systems. Our methods have worked flawlessly with DOS 3.3 through 5.0 and have not interfered with the operation of programs as diverse as WordPerfect, Lotus 1-2-3, Quattro Pro, and dBase. Our testing is by no means exhaustive, however, so it is conceivable that problems may arise with older versions of DOS,

***Editor's Note:** The techniques described here let you reprogram your F1, F2, F3 keys—keys to which DOS assigns command-editing and -recall capabilities. If you redefine these keys and later wish to restore their original capabilities, you can do so simply by rebooting your computer or—if you reprogram the keys through a batch file—by removing the relevant line or lines from your batch file and then rebooting.*

older hardware, or certain software.

WORKING WITH CONFIG.SYS

CONFIG.SYS is a text file that DOS looks for each time it starts up your computer. The statements listed in CONFIG.SYS provide DOS with important setup information, such as the types of devices—a mouse and a RAM (virtual) drive, for instance—that you want installed. Many computer manufacturers and dealers set up a machine's CONFIG.SYS file before delivering the computer to a customer.

CONFIG.SYS resides in the root directory of the boot drive. These days, that means you usually can locate it by typing *DIR C:* at the DOS prompt. If you don't find a copy of CONFIG.SYS, you can create one. If you plan on typing only a few lines, the quickest way to create the file may be to work from the DOS prompt in your root directory. Simply type:

```
COPY CON CONFIG.SYS
```

to tell DOS you are about to type a file named CONFIG.SYS at the console (CON), and press Enter. Then type the other lines you need, pressing Enter after each one. For the purposes of this article, you need only three additional commands:

```
FILES=20
BUFFERS=20
DEVICE=C:\DOS\ANSI.SYS
```

To save the file as CONFIG.SYS and return to the DOS prompt, press Ctrl-Z.

If CONFIG.SYS already exists, you can make the appropriate changes by calling up the file in a word processor that saves in ASCII format, bringing it up in DOS's Edlin line editor, or by calling it up in DOS 5.0's Edit full-screen editor. Make the necessary changes and additions (be sure to include the DEVICE=ANSI.SYS line), and save the file.

To have the new CONFIG.SYS settings take effect, you must reboot your computer.

—A.P. & A.P.

Setting the Stage

Before you begin, you must make provisions to load DOS's ANSI.SYS device driver program into memory each time you start up your computer. This requires adding a DEVICE statement to your system's CONFIG.SYS file. (See the sidebar "Working With CONFIG.SYS," if you are unfamiliar with this start-up file.) If ANSI.SYS resides in a subdirectory called \DOS that is one level below your hard drive's root directory, here is the proper command to accomplish this:

```
DEVICE=C:\DOS\ANSI.SYS
```

While you are in CONFIG.SYS, check to make sure that the file includes a BUFFERS and FILES statement and that at least 20 buffers and 20 files are allocated. Once you know that ANSI.SYS or a compatible driver, such as FANSI.SYS or SIZZLE.SYS, is in your CONFIG.SYS file, reboot your computer to load the device driver.

The secret to programming your system's function keys is using the DOS PROMPT statement and the "escape" sequence $E (or $e) followed by the appropriate ASCII key code. The syntax for the command is:

```
PROMPT $E[ASCII key
 code;"command$_"p
```

This statement may seem intimidating at first, but it's quite manageable if you look at a real-life example and break it down into understandable segments. The following one-line statement, for instance, redefines the F4 key to execute a DIR command:

```
PROMPT $E[0;62;
 "DIR$_"p
```

It comprises seven parts:

- PROMPT is the DOS command for changing the system prompt.
- $E denotes the escape character.
- The left bracket ([) separates the function-key code from the escape character.

TIP THE KEY TO MORE DOSKEY

Most people realize that the DOS 5.0's DOSKEY utility maintains a history of previously executed commands. What you may not know is that, by default, DOSKEY sets up only a 512-byte command buffer—enough to store approximately five macros and 10 command lines. Simply adding the following BUFSIZE parameter to your DOSKEY command will double the size of the DOSKEY command buffer (to 1KB) and, therefore, will increase the number of macros and command lines it can store:

```
DOSKEY /BUFSIZE=1024
```

—J.N.

ASCII KEY CODES

	ASCII Values			
Function Key	**Alone**	**Shift+Key**	**Control+Key**	**Alt+Key**
F1	0;59	0;84	0;94	0;104
F2	0;60	0;85	0;95	0;105
F3	0;61	0;86	0;96	0;106
F4	0;62	0;87	0;97	0;107
F5	0;63	0;88	0;98	0;108
F6	0;64	0;89	0;99	0;109
F7	0;65	0;90	0;100	0;110
F8	0;66	0;91	0;101	0;111
F9	0;67	0;92	0;102	0;112
F10	0;68	0;93	0;103	0;113
F11	0;133	0;135	0;137	0;139
F12	0;134	0;136	0;138	0;140

- 0;62 is the ASCII equivalent for the F4 key.
- The second semicolon (;) separates the ASCII key code from the command to be executed.
- "DIR$_" is a DOS directory command followed by a carriage return ($_). Both the command and the carriage-return designator must be surrounded by quotation marks.
- The final p marks the end of the prompt. (Note that this letter must be a lowercase character.)

If you type this command correctly, you can display a directory by pressing the F4 key. Your original DOS prompt will disappear, but you can recall it, if you like, simply by issuing another PROMPT statement (such as PROMPT PG).

If you don't use an unembellished DIR command very often, don't worry. You can include command switches in the PROMPT statement. For instance, replacing "DIR$_" with "DIR /P$_" in the above example reprograms F4 to display a directory listing page by page.

To program function-key combinations, such as Shift-F1, you must use a different set of ASCII codes. (See the table, "ASCII Key Codes," for the values of function-key sequences involving the function keys and Shift, Ctrl, and Alt. Refer to your DOS manual to find the ASCII key codes for additional keys, such as PgUp, PgDn, Home, and End.) The following command, for example, redefines the Shift-F1 combination to clear the screen (CLS):

```
PROMPT $E[0;84;"CLS$_"p
```

Thus far, all the examples we've provided automatically perform a carriage return for you. You may, however, encounter certain situations under which you don't want to execute a command immediately. The solution is as simple as omitting $_ from your command. The following prompt, for instance, assigns the FORMAT command to the F5 key:

TIP BACKGROUND PRINTING IN DOS

If you're new to DOS, you may not realize that the PRINT command lets you print one or more text files while you continue to use your computer to run other DOS commands. You can format a disk, for example, while printing text files in the background.

PRINT's syntax is straightforward. For instance, to print out AUTOEXEC.BAT, which is a batch file and therefore qualifies as a text file, you would type:

```
PRINT C:\AUTOEXEC.BAT
```

(Naturally, you must specify a complete pathname for the PRINT.COM utility and the correct drive for AUTOEXEC.BAT.) PRINT first asks for the name of the printer port you are using, but it does so with the usual jargon:

```
Name of list device [PRN]:
```

If you are using a printer attached to a parallel port (which covers most of us), just press Enter. If your printer is attached to a serial port, you'll have to type the name of the port; COM1 is the most likely answer. If you respond correctly and your printer is on, you then receive a message telling you that your AUTOEXEC.BAT is being printed.

If you want to print a long list of text files in a word processing directory, you can do so easily by using the wildcard character * or ?. PRINT will accept the files, tossing them all into a print queue. If, for instance, you use Professional Write in a directory called \PW and you type:

```
PRINT C:\PW\*.TXT
```

DOS will load into the queue as many TXT files in that directory as the print buffer can accommodate.

—Gloria Short

```
PROMPT $E[0;63;"FORMAT "p
```

Because it doesn't execute a carriage return, the command followed by a space (because a space appears after the FORMAT command) is displayed when you press F5. You then can specify the letter of the drive containing the disk to be formatted and press Enter to tell DOS to process the command.

Program Listing. DEFKEY.BAT assigns tasks to a series of function-key sequences. Using a word processor that saves in ASCII format, type in and save the program. To run the program and define the keys, type in DEFKEY at the DOS prompt and press Enter, or invoke the batch file from within your AUTOEXEC.BAT file. Please note that some of the commands mentioned are specific to DOS 5.0.

```
CLS
PROMPT $E[0;59;"HELP$_"p          'F1 = Display DOS's help screen
PROMPT $E[0;60;"CLS$_"p           'F2 = Clear the screen
PROMPT $E[0;61;"COPY "p           'F3 = Issue COPY
PROMPT $E[0;62;"XCOPY "p          'F4 = Issue XCOPY
PROMPT $E[0;63;"DISKCOPY "p       'F5 = Issue DISKCOPY
PROMPT $E[0;64;"RENAME "p         'F6 = Issue RENAME
PROMPT $E[0;65;"ERASE "p          'F7 = Issue ERASE
PROMPT $E[0;66;"FORMAT "p         'F8 = Issue FORMAT
PROMPT $E[0;67;"BACKUP "p         'F9 = Issue BACKUP
PROMPT $E[0;68;"RESTORE "p        'F10 = Issue RESTORE
PROMPT $E[0;133;"ATTRIB "p        'F11 = Change the attribute of a file
PROMPT $E[0;134;"CHKDSK C:$_"p    'F12 = Check the integrity of a disk
PROMPT $E[0;84;"HELP/?$_"p        'SHIFT-F1 = Display help on HELP
PROMPT $E[0;85;"CLS/?$_"p         'SHIFT-F2 = Display help on CLS
PROMPT $E[0;86;"COPY/?$_"p        'SHIFT-F3 = Display help on COPY
PROMPT $E[0;87;"XCOPY/?$_"p       'SHIFT-F4 = Display help on XCOPY
PROMPT $E[0;88;"DISKCOPY/?$_"p    'SHIFT-F5 = Display help on DISKCOPY
PROMPT $E[0;89;"RENAME/?$_"p      'SHIFT-F6 = Display help on RENAME
PROMPT $E[0;90;"ERASE/?$_"p       'SHIFT-F7 = Display help on ERASE
PROMPT $E[0;91;"FORMAT/?$_"p      'SHIFT-F8 = Display help on FORMAT
PROMPT $E[0;92;"BACKUP/?$_"p      'SHIFT-F9 = Display help on BACKUP
PROMPT $E[0;93;"RESTORE/?$_"p     'SHIFT-F10 = Display help on RESTORE
PROMPT $E[0,135;"ATTRIB/?$_"p     'SHIFT-F11 = Display help on ATTRIB
PROMPT $E[0,136;"CHKDSK/?$_"p     'SHIFT-F12 = Display help on CHKDSK
PROMPT $P$G
CLS
```

End

Prompts by the Batch

You probably already are brimming with ideas about how you might apply this technique to your computing routine. Unless you use a batch file to automate the procedure for invoking PROMPT statements, however, you'll find it inconvenient to take advantage of your ability to program function keys. Fortunately, creating a batch file to this is easy. Using a word processor or any other editor that saves in ASCII format, you simply type and save your batch file. The general format for such a batch program might look like this:

```
CLS
PROMPT
$E[0;59;"DIR$_"p
PROMPT $E[0;84;"CLS$_"p
PROMPT $E[0;63;"FORMAT "p
.
REM Additional function key
REM definitions go here.
.
PROMPT $P$G
```

The first statement clears the screen; the function-key definitions come next. Finally, the last statement resets the system prompt. Note that the program does not, however, include an ECHO OFF statement, because the PROMPT statements must be output to the screen for the program to work properly. The Program Listing, DEFKEY.BAT, provides a more extensive list of commands.

If you want to enable your function-key definitions each time you boot up, simply insert a line in AUTOEXEC.BAT that calls the batch file by name. To call the DEFKEY.BAT, for instance, you would add the following line to AUTOEXEC.BAT:

```
CALL DEFKEY
```

The CALL statement is available in DOS 3.3 and later, and it allows you to run another batch file without stopping the calling batch file.

Of course, these examples just scratch the surface of what you can accomplish by programming function-key sequences. You also might try developing a set of function-key definitions that load your most frequently used programs, for instance, using F4 to load WordPerfect, F5 to load Lotus 1-2-3, and so on. But don't stop there. Feel free to experiment; your efforts will be rewarded. ■

Free-lance editors Antonio and Alfonso De Pasquale live in metropolitan Boston.

How to Beat Modem Anxiety

By learning a few terms and concepts, you can overcome your fear of telecommunications.

by Jack Nimersheim

Many PC users are intimidated by telecommunications. Yes, they know the benefits of retrieving data from on-line services and bulletin board systems (BBSes). But they are overwhelmed by the many complex procedures and confusing terms that stand in their way.

If you're nodding your head right now, take heart. You can eliminate most of your anxiety just by understanding some of the concepts behind telecommunications. Then, with a modem and the right software, a few simple steps will get you on line.

The idea behind telecommunications is simple enough. You link your computer to another over a telephone line, then send and retrieve data. What complicates the process is that computers and phone lines are incompatible. The PC is a digital machine, while phones use analog signals. That is, while the PC sees data as a series of electronic 1's and zeros, telephone lines carry only continuous sound waves.

To fix this problem, the transmitting PC must convert its 1's and zeros into sound waves. The receiving PC then must change these waves back into 1's and zeros. This is what the modem is for. Each PC has one of these devices between it and the phone line, translating the incoming and outgoing information.

To work, a modem needs instructions from you on what to do. You use telecommunications software to issue commands. The software also lets you read, store, and otherwise manage incoming data.

Most modems come with a communications program, usually a simple shareware package. While this will get you up and running, some commercial programs have more features and are easier to use.

Older software forced the user to memorize arcane commands to do even simple tasks. The ATDT command, for example, told the modem to be ready to dial a number and informed it that it was attached to a touch-tone system. Today's communications software uses pull-down menus or some other command structure. The program then translates your request into the appropriate commands.

The sidebar "A Software Sampler: PC Communications Programs" profiles several popular MS-DOS communications programs.

Making a Modem Work

Hooking up a modem is easy. If you have an external modem, you just plug it into an unused communications port—or COM port—in the rear of your PC. If you use an internal modem card, you snap it into an unused card slot inside your machine. The COM port, which actually is a serial port by another name, is an electronic doorway that lets signals pass between your modem and PC.

The most difficult task is picking the right COM port. Just as two people have trouble walking through the same door at the same time, two peripherals can't use the same COM port. If you have a mouse attached to

THE LANGUAGE OF GOING ON-LINE

Analog signals: The type of tones normally transmitted over standard telephone lines. A modem converts the digital signals generated by your PC into analog tones during an on-line session.

AT command set: A series of commands used to manage a modem. This command set derives its name from the fact that you start commands with the letters *AT* to get the attention of your modem. First popularized by Hayes modems, the AT command set has evolved into a standard for PC telecommunications.

Auto-answer: Your modem's state when it has been instructed to answer incoming calls automatically and establish a connection with the computer that's calling.

Baud rate: The number of times a modem changes the frequency of its carrier signal. Although this term often is used interchangeably with bits per second (see below)—the speed of a modem—they are not the same. A modem's baud rate refers to the number of electronic events processed, or signal changes in one second within the modem. These events can process more than one bit of data.

BBS: An acronym for *bulletin board system.* As a rule, BBSes are open systems; that is, they are available to anyone who calls them. Thousands of BBSes operate throughout the United States.

Bit: Contraction for binary digit, the smallest information unit a computer can identify.

Bps: An abbreviation for *bits per second.* This is the number of bits transferred every second during an on-line session. The bps measurement can differ from baud rate, since multiple bits often can be transmitted with a single state change.

Communications port: Also called a COM port, this actually is a serial port installed on your PC. For your communications software to work properly, you must specify the COM port to which your modem is attached .

Communications parameters: The rules two computers use to communicate with one another during a connection. These settings include the data rate (see bps above), parity, stop bits, and file transfer protocol (see below).

Communications software: A program that lets your PC use a modem and standard telephone line to contact a remote system.

Data word: An organized group of bits. Individual data words are organized and separated using start bits and stop bits. The most common settings used for PC communications are 7- and 8-bit data words.

Digital signal: The type of signal your PC generates. Generally, a digital signal is represented by either an "on" or "off" state, a zero or 1, or some such notation. A modem converts the analog tones coming over a phone line into the digital signals your PC needs during an on-line session.

Download: The process of transferring files and information from a remote system to your PC.

E-mail: An abbreviation for *electronic mail*, messages (usually text files) sent via modem to another computer, where they are stored until the recipient picks them up.

Emulation: The process of configuring your PC to resemble a specific terminal type. Many communications programs, for example, let you emulate a Digital Equipment Corp. VT100 terminal so you can communicate with a DEC mainframe.

Modem: A device used to convert analog tones to digital signals, and vice versa, during an on-line session. A contraction for *modulate/demodulate*, the technical term for this process. Modems can be external or internal. An external modem is a separate piece of equipment connected to a phone line and your PC's serial port. An internal modem is a board that you install in an expansion slot inside your PC. The board carries its own jack for attaching the phone line directly to the computer.

Parity bit: A bit added to a character and used for catching transmission errors. The parity bit tells whether the number of 1 bits in the character is even or odd. The receiving modem counts the 1 bits and checks its figure against the parity bit. If the two don't match, the modem knows the transmission was unsuccessful.

Protocol: The rules two computers use to initiate and verify file transfers. A number of protocols exist—Xmodem, Ymodem, Kermit—and are available in most commercial software programs.

Script: A series of preset keystrokes that trigger commands or menu selections written in a simple programming language found in many commercial software packages. A script will execute in sequence when you dial up a bulletin board or on-line service. For example, a script can instruct the computer to dial MCI mail and give your user name, password, and the command to scan your in-box. It can download your mail and exit the mail system, all within a few seconds.

SIG: An abbreviation for *special-interest group.* Many on-line commercial services and most BBSes provide areas such as these, where those interested in a specific topic (aviation, photography, stamp collecting) can converse or exchange ideas and messages.

Start bit: A bit that marks the beginning of a character.

Stop bit: A data bit that marks the end of a character.

Sysop: An abbreviation for *system operator*, the person responsible for maintaining a BBS or a specific special-interest area within a BBS or commercial service.

Upload: The process of transferring files and information from your PC to a remote system.

COM1 on your computer, for example, you'll have to pick a different COM port for your modem.

If you have an external modem, just hook it to an unused serial port. If you use an internal modem, its manual will explain how to pick the right COM setting.

When you install a communications program, you must tell it which COM port to use. If your modem does not respond to commands sent from your communications program, the cause usually is an incorrect COM port setting. Changing the setting should correct the problem.

Transmission Rates And Other Esoterica

Before you call another computer, you must tell your software what parameters to use. These are settings that control such factors as transmission speed and the size of each information packet. Your parameters must match those of the computer to which you're connecting.

The four parameters you need to know are transmission rate, parity, data-word size, and stop bits. The transmission rate, measured in bits per second (bps) and sometimes incorrectly referred to as baud rate, is

A SOFTWARE SAMPLER: PC COMMUNICATIONS PROGRAMS

Choosing a communications program is a lot like buying a car. You'll be spending quite a lot of time cruising the on-line highways with your program, so you should make sure it has the options you want.

Newcomers to telecommunications should look for features that make a program easier to use. These include pull-down menus, a well-designed calling directory, and a "learn" function that helps create scripts (see the glossary).

Experienced telecommunicators favor power over convenience. For them, a comprehensive script language that can automate complicated activities is a must. Terminal emulation (which permits your PC to mimic another computer, particularly a mainframe or minicomputer) and multiple file transfer protocols also become more important the higher up the on-line ladder you climb.

If possible, ask to see a demonstration of the communications program you're considering before buying it. This will let you determine if its features match your needs. Here are a few of the more popular packages on the market:

ProComm Plus, $129
Datastorm Technologies Inc.
P.O. Box 1471
Columbia, MO 65205
800-326-4999
314-443-3282

ProComm Plus became one of the top-selling DOS communications programs by combining power with ease of use. Its Dialing Directory can accommodate as many as 200 phone numbers. Calling a system recorded in the dialing directory is a simple matter of highlighting its entry and pressing the Enter key.

ProComm Plus includes a powerful script language called Aspect. By attaching an Aspect script to a Dialing Directory entry, you can reduce most of your activities to a single-key operation.

ProComm Plus lets you select from among 33 different types of terminal emulation, including several IBM, AT&T, Wyse, and Digital Equipment Corp. mainframe and minicomputer terminals. You can choose from 15 standard protocols for file transfers. Popular protocols supported include Xmodem, Ymodem, Kermit, ASCII, and CompuServe B+.

ProComm, which originally was a shareware program, still is available through that channel. The shareware version, simply called ProComm, offers fewer file-transfer protocols and doesn't emulate as many terminals. Nor does it include the Aspect script language. It is, however, an ideal beginner's program. Furthermore, everything you learn using ProComm carries over to the Plus version.

Datastorm Technologies has not ignored Windows users, either. The company also markets ProComm Plus for Windows ($149).

MicroPhone II, $129
Software Ventures Corp.
2907 Claremont Ave., Suite 220
Berkeley, CA 94705
510-644-3232

MicroPhone II runs under Windows, a graphical user interface that simplifies many PC activities.

One of the most impressive features of MicroPhone II is its script language. Creating scripts with MicroPhone II is automated almost totally. Whereas most communications programs force you to create and edit scripts with a script editor—a procedure similar to creating MS-DOS batch files—MicroPhone II includes a special module that lets you interactively pick the operations you want your script to perform.

The MicroPhone II package ships with preprogrammed scripts for several popular commercial on-line information systems, including CompuServe, the Byte Information Exchange (BIX), Dow Jones News/Retrieval, MCI Mail, and GEnie.

Crosstalk Communicator, $99
Digital Communications
Associates Inc.
1000 Alderman Dr.
Alpharetta, GA 30202
800-348-3221
404-442-4000

Crosstalk Communicator is a scaled-down version of the successful Crosstalk Mk.4 program. Communicator ships with 10 preconfigured dialing directory entries for such services as CompuServe and Dow Jones.

The program's file transfer protocol options include CompuServe B, Xmodem, Ymodem, and Zmodem. Communicator offers equally impressive terminal emulation options, including Digital Equipment Corp.'s VT52 and VT100 protocols, the American National Standards Institute (ANSI) protocol, teletype (TTY) protocols, and IBM's 3101 protocol.

At $99, Crosstalk Communicator is a bargain. In fact, it may be the perfect program for someone just getting started in telecommunications.

—J.N.

AN ON-LINE SAMPLER

Looking for a place to call? Here are a few candidates.

For services you connect to directly, I've included the appropriate telephone number and parameter settings. In other cases, primarily commercial information and e-mail services, I've listed a voice number you can use to get more information. If you subscribe to one of these services, the company's representative will tell you how to contact it with your modem.

Commercial Information Services

CompuServe Inc.
5000 Arlington Centre Blvd.
Columbus, OH 43220
800-848-8199
614-457-0802 in Ohio and Canada

The most successful commercial service currently in business, CompuServe claims more than 1 million subscribers and provides access to over 1,700 databases and special-interest forums.

Its monthly fee is $7.95 for basic services. Other services are billed by the hour, with prices ranging from $6.30 to $22.80 per hour, depending on the transmission rate and service you use.

CompuServe's logical menu structure eases navigating—even for first-time users. Many computer hardware and software manufacturers, including major companies, such as Microsoft Corp. and IBM, provide on-line technical support and advice through CompuServe. Other special-interest groups cover topics ranging from comic books to stamp collecting.

If there's a generic information service, this is it. CompuServe is probably the best place to start, if you're just starting out.

Delphi
General Videotex Corp.
1030 Massachusetts Ave.
Cambridge, MA 02138
800-695-4005
617-491-3393 in Massachusetts

Delphi offers about 75 special-interest areas,including sections on personal computers, writing, religion, and new age philosophy.

When you sign up, you have two billing options, the 10/4 Plan and the 20/20 Advantage Plan. The less expensive of the two, the 10/4 Plan, charges $10 for your first four hours of off-peak, on-line time each month; each off-peak hour after that is $4. Those who plan to spend more time on line might want to sign up for the 20/20 Advantage Plan, which allows 20 hours per month of off-peak usage for $20; each additional off-peak hour costs $1.80. Peak-time (7 a.m. to 7 p.m., Monday through Friday) usage for both plans is $9 per hour.

For on-line access to Delphi and subscription information about this service, Set your modem to 300, 1200, or 2400 bps; 8 bits, no parity, and 1 stop bit; and call 800-365-4636 or 800-695-4002. When you're connected, press Enter once or twice to "wake up" the Delphi system. At the Username prompt, type `JOINDELPHI` and press Enter. At the Password prompt, type `SIGNUP` and press Enter.

Although Delphi's subscription base is not as large as CompuServe's, its members tend to be more serious about their on-line activities. The Science Fiction SIG, for example, includes more professional writers than any other on-line service I've used. Every Wednesday night, several of these professionals participate in an open conference that also includes editors, fans, and neophyte writers seeking advice.

The choice of whether to subscribe is yours. If you do subscribe and are dissatisfied, the company offers a money-back guarantee.

Dow Jones News/Retrieval
P.O. Box 300
Princeton, NJ 08543
800-522-3567
609-452-1511

If business is your game, then you should investigate Dow Jones. The service gives you fast access to up-to-the-minute stock quotes for both domestic and international exchanges, 24 hours a day. Dow Jones charges a $29.95 sign-up fee for standard membership. After that, several options are available, including a monthly flat fee of $25 for basic operations and per-minute billing (60 cents to $2.16) for other services.

Dow Jones lets you download transcripts of articles appearing in *The Wall Street Journal*. It also offers access to a variety of company and industrial databases that resemble on-line newsletters. These include Worldscope, Zacks Corporate Earnings Estimator, and Duns Financial Records.

GEnie
GEIS Division of General Electric
401 N. Washington St.
Rockville, MD 20850
800-638-9636

If you think CompuServe or Delphi is inexpensive, wait until you hear about GEnie. This popular commercial information service recently reduced its monthly fee to a mere $4.95, for unlimited access (during off-peak hours) to more than 100 basic products and services. GEnie's rock-bottom monthly fee makes it an attractive alternative for anyone wanting to test a commercial on-line service. For a little investment, you can learn a lot. Be aware, however, that the access-fee clock ($6 per hour) starts ticking for GEnie's Value Services, which

the speed at which you send your data. The parity setting tells the software whether you want error checking and, if so, what kind. The data-word size is the number of bits in each packet of data you're sending. The stop bit is a bit that marks the end of a character. (See the glossary, "The Language of Going On Line," for more detailed definitions.)

For example, suppose you want to call a BBS that asks you to set your parameters at 2400, N, 8, 1. This notation means that the system sends and receives data at 2400 bps with no parity, uses 8 data bits per word, and has 1 stop bit. Prior to calling that BBS, you need to use the com-include such popular activities as real-time conferencing and games.

GEnie's forte is setting up direct communications with large groups of people through on-line messaging. Individual sections cover such topics as investments, employment opportunities, and operating a small business. Start a message thread in any of these areas and, within a few days, you'll be able to sign back on and find more advice and opinions than you can handle.

GEnie's most obvious weakness is its convoluted and often inconsistent menu structure. Command shortcuts are available, but they are obtuse. Individual sections are identified by page numbers or cryptic abbreviations. GEnie is definitely not the service for someone who's a bit intimidated about going on line.

Prodigy
445 Hamilton Ave.
White Plains, NY 10601
800-776-3449

To access Prodigy, you need special software, which you can buy at computer stores. (By calling Prodigy's toll-free number, you can receive information on the location of a store in your area.) In addition, the service charges a monthly fee of $12.95.

Prodigy offers more consumer oriented-options than other services. For example, a product index provides news and objective reviews on a wide range of goods and services, as well as current product ratings from *Consumer Reports* magazine. Prodigy also offers current stock market information, provided by Dow Jones.

The service's visually oriented user interface is easier to use than the cryptic command menus most others provide. Thus, it's a natural choice for the non-technical neophyte just getting into modem communications.

Prodigy is one of the larger services, claiming more than 1.5 million subscribers.

Electronic Mail Services

AT&T Mail
Easylink Services
400 Interpace Parkway
Parsippany, NJ 07054
800-242-6005

This is AT&T. What more need be said? AT&T Mail charges a monthly fee of $3. In addition, you pay for each message sent, with price dependent on message length. Like many e-mail services, AT&T offers a toll-free number for connection, thus saving you long-distance charges.

MCI Mail
1133 19th St. N.W., 7th Floor
Washington, DC 20036
800-444-6245

And in this corner...MCI. Yes, the two biggest telephone companies also are squaring off against one another in the e-mail market. MCI charges a $35 annual fee—that's nearly the same as AT&T's 12-month cost. It's per-character charge for messages, however, is lower, especially for longer missives.

Local BBSes

You'll find thousands of local BBSes out there. The differences between them are almost as legion as their numbers. Most, however, include a section for exchanging messages with other callers. BBSes also are a good place to find shareware and public-domain programs. Finally, it's not unusual for one BBS to provide listings of other BBSes, not only in the immediate area, but throughout the country.

Try signing on to one of the following BBSes. Explore it. See if other boards are located in your area. The only way to find out how much fun on-line communication can be is to go on line. Try it—you'll probably like it.

Adairville, Ga.
Warp Factor BBS
706-773-7966
300 to 9600 bps, N, 8, 1

If you're interested in "Star Trek," try this one; it contains several discussion areas about both the original and current shows.

Cincinnati, Ohio
The Multisystem TBBS
513-231-7013
300 to 38,400 bps, N, 8, 1

This BBS contains more than 10,000 shareware programs. Two lines support on-line conferencing and games.

Frederick, Md.
Advanced Data Services Inc.
301-695-9116
300 to 19,200 bps, N, 8, 1

This 25-line PC board claims to be the largest in the Northeast—although I'm not sure Maryland is in the Northeast. Its 5.5GB collection includes 2,400 PC-SIG files and 1GB of adult files.

Indianapolis, Ind.
Indiana BBS
317-297-4224
300 to 2400 bps, N, 8, 1

The Indiana BBS includes an open forum on home learning and schooling, plus information on railroading.

West Hartford, Conn.
Bruce's Bar & Grill
203-236-3761
300 to 2400 bps, N, 8, 1

This is a social SIG for denizens of Bruce and Rachel Lomasky's bar. Mostly locals gather here, but if you've ever wondered what's to do in West Hartford, here's a clever way to find out. You'll find a wine cellar, a sports network, and the usual chat SIG. To enter the over-21 section, however, you'll have to send Bruce proof of your age.

—J.N.

mands in your communications software to specify these settings.

The two most common parameters used for PC communications are 2400, N, 8, 1 and 2400, E, 7, 1. Many 2400-bps systems, however, also let you use a 1200-bps modem.

If you connect to a remote system and see an odd assortment of characters and symbols—what we high-tech computer types commonly refer to as "garbage"—the most likely culprit is a parameter mismatch. Don't panic—just change parameters.

Return to Sender

Assuming your parameters are correct and you have a good connection, you should have no problem reading data and sending simple messages. But to get the most out of information systems, you eventually will want to download—that is, receive and save—files. You probably also will want to upload, or send, files. To do so, both computers must agree on how to verify that the data is being sent and received properly. They accomplish this with something called a file-transfer protocol.

To understand how protocols function, consider the rules of grammar. When you see a space following a word, you immediately know the word has ended. This article would be extremely hard to read if everythinglookedlikethis.

Protocols apply similar rules. They organize a file's contents into structured data blocks identifying the size of individual blocks in the file, their position in relation to the entire file, how blocks are separated, and whether and what kind of error checking will be used. When both computers agree on exactly how the data in a file will be organized, they can coordinate a file exchange successfully.

Some popular communications protocols you will encounter on line are ASCII, Xmodem, Ymodem, and Kermit. All have their merits. As a rule, you should use the ASCII protocol for transferring text files. If you are transferring data and program files—that is, binary files—Xmodem is the most universal protocol. Whatever you use, remember: Both computers must use the same protocol to transfer a file.

On-line Options

Okay, so you have your modem and communications program set up and running properly. Where do you go from here? You go on line, of course.

For simplicity, let's break your online options down into three major categories: e-mail services, commercial information services, and local BBSes. The first two usually have a connect charge or, in the case of some e-mail services, a per-message fee. BBSes are less formal and often are free.

The sidebar "An On-Line Sampler" lists representative services and the connect procedures and communications parameter settings for each. BBSes are a good place to start. After providing some basic information about yourself, you usually can use them the first time you call. Commercial services, on the other hand, usually require you to become a subscriber.

Once you do connect with a remote system, you'll be able to perform different kinds of on-line activities, depending on which service you call. These include:

- Electronic messaging or e-mail (available through e-mail services, commercial services, BBSes)
- Real-time conferencing (commercial services, BBSes)
- File transfers (commercial services, BBSes)
- Research and information gathering (commercial services)
- Electronic transactions; e.g., banking, shopping (commercial services)

So hook up the modem, install your new communications software, and get started. There's a whole world waiting out there for you. Literally. ■

Jack Nimersheim, a computer journalist and free-lance writer, has been using and writing about personal computers for more than 10 years. Some of the material in this article was excerpted from his First Book of Modem Communications *(Sams/Prentice-Hall).*

TIP TYPE IT PDQ!

Despite the sophisticated software available for your computer, you may find times when it isn't worth the trouble to load your word processor simply to address an envelope or type a quick note. If pencil and paper just won't do the job, the following short batch file lets you use your printer as a rudimentary typewriter:

```
@ECHO OFF
CLS
ECHO  Press Enter to end each line.
ECHO  Press F6 and Enter to quit.
COPY CON:PRN
ECHO ^L >PRN
CLS
```

Save the file as TYPER.BAT or something similar to avoid confusion with DOS's TYPE command. (In the sixth line, type the ECHO command, hold down the Alt key while typing the sequence 012 on the numeric keypad, release the Alt key, and press Enter.)

The COPY CON command in the fifth line lets you type several lines of text from the keyboard. When you press F6 followed by Enter, DOS sends the text to your printer. The ^L sequence in the next line is required for page printers (such as laser printers) that need a form-feed character (ASCII 12) to print and eject a page. The DOS "output" redirection symbol (>) adds the ASCII form-feed character to the end of the text you've typed.

—J.W.

DR DOS 6.0: A Most Worthy Opponent

With a number of unique attributes—and some embellishments to the way MS-DOS does business—Digital Research's operating system presents a solid case for leaving MS-DOS behind.

by John Wolfskill

Last year, Digital Research's DR DOS 6.0 was hailed widely as a breakthrough alternative to MS-DOS, the operating system standard for IBM and compatible PCs. Certainly, Novell Inc., maker of NetWare networking software, believed in the product so much, that it bought Digital Research. But is this upstart operating system a serious contender in an industry where Microsoft has always been the only game in town? In my opinion, yes.

For openers, set aside any suspicions you may have about compatibility. DR DOS 6.0 is compatible with every MS-DOS application—including Windows. You'll also find extra value in DR DOS 6.0. Digital Research included powerful utilities in its operating system: the ability to switch from task to task, 80386 memory management, and a hard disk optimizer, to name a few. But the real reason many PC users are switching to DR DOS is to take advantage of its unique features and commands, among them an expanded batch file language.

If you're on the outside of DR DOS looking in or have purchased the product but haven't yet had an opportunity to explore its many nooks and crannies, this quick look at DR DOS 6.0 is for you.

File Recovery Made Easy

I'll start with DR DOS's UNDELETE command, because it offers you the peace of mind any operating system should. It's a shame that Microsoft waited until Version 5.0 to bring such an essential feature to DOS. In all fairness, however, DR DOS's UNDELETE is no better or worse than the UNDELETE module offered in DOS 5.0.

The DR DOS UNDELETE command recovers files or groups of files accidentally deleted during a work session. A related command, DELWATCH, helps UNDELETE do its job more efficiently. DELWATCH is a 6K terminate-and-stay-resident (TSR) program that—once installed—hangs out in memory, saving information about the names and locations of files you've marked for deletion and making it available to UNDELETE.

With DELWATCH on guard, deleted files only appear to be expunged. DR DOS marks the file as one whose deletion is pending, without erasing it from your disk. This means that you can change your mind later and revive the file using the UNDELETE command. DELWATCH loads into upper memory when it is available. However, you can force it to load into conventional (low) memory by specifying the /ML switch:

```
DELWATCH C: /ML
```

By default, DELWATCH limits to 200 the number of files whose deletion is pending. If you prefer, you can set a higher number by specifying the command line switch /F:65535 to allow the maximum number of files or by using /F:ALL to specify all files, provided you don't exceed the maximum. When you specify either of these switches, DELWATCH guarantees all files pending deletion can be recovered. For example this command:

```
DELWATCH C: /F:ALL
```

loads DELWATCH into memory, where it begins to track the files you delete from drive C. The /F:ALL switch tells DELWATCH to track the status of all deleted files, whether you decide to purge one or 1,000 files.

Although DELWATCH ensures that all files marked for pending deletion can be recovered safely, it also reduces the amount of memory available to your programs. And, since DELWATCH is a TSR program, it may not work with applications that "steal" the software interrupt vectors (memory-address pointers) that TSRs need to function properly. An example are applications that run under early versions of Microsoft Windows.

For DR DOS users with a limited amount of RAM or a working environment that doesn't cooperate with TSRs, the DISKMAP command provides an alternative to DELWATCH. It requires no memory and doesn't operate as a TSR. Although DISKMAP isn't as efficient at tracking files, it provides a way to save information about deleted files that helps UNDELETE recover them.

Unlike DELWATCH, DISKMAP doesn't save detailed information about deleted files. Instead, it saves the most recent copy of your file allocation table, or FAT, as it's commonly called. The FAT is a file map stored at the start of your hard or floppy disk. DR DOS uses the FAT to locate the files stored on the disk. During file recovery operations, DISKMAP can give UNDELETE the original location of a deleted file, as long as DR DOS hasn't reallocated its disk space to a new file.

$X Marks the Spot

Ideally, you should call DISKMAP to obtain a "snapshot" of the FAT index after every file you delete. Since DISKMAP doesn't lurk in the background awaiting your signal to "pop up," you must find a way to activate it at the right times. Fortunately, the DR DOS PROMPT command provides a clever built-in mechanism to do the job.

PROMPT lets you modify the way DR DOS displays its system prompt. You can set the prompt to display any valid ASCII character or character string (such as C:>). Among the unique characters that PROMPT recognizes is $X, which causes PROMPT to execute a command (or even run a program) each time it returns you to the system prompt. Although $X isn't designed to execute DISKMAP, it offers a way to trigger DISKMAP each time you return to the system prompt.

To use PROMPT for this purpose, add the following lines to your AUTOEXEC.BAT file:

```
SET PEXEC=C:\DRDOS\DISKMAP C:
PROMPT $X
```

TIP LOOK: NO HANDS

If you've discovered the advantages of using batch files to automate DOS tasks, you've also discovered that many programs frequently ask for keyboard input—a snag that can stop a batch file dead in its tracks.

Fortunately, the response expected by many programs is predictable: They ask you to confirm or deny an action they are about to perform. You can create a data file that lets your batch files proceed, without requiring you to type in a response from the keyboard. A typical data file of this sort might look like the following:

```
@ECHO OFF
ECHO Y > YES.DAT
MYFILE < YES.DAT
DEL YES.DAT
```

In this example, an application program called MYFILE.EXE asks that you type *Y* or *N* and press Enter.

The second line of the batch file uses the DOS output redirection symbol (>) to place the character *Y* and a carriage return in a file called YES.DAT. The third line reverses this process by employing the input redirection symbol (<) to place the *Y* character stored in YES.DAT in the keyboard buffer just before the program MYFILE asks for a keyboard response. The last line simply erases YES.DAT after it has served its purpose.

—J.W.

$X causes PROMPT to look into the DR DOS environment space for a string of text characters you've assigned to the environment variable PEXEC. You can use the SET command to assign DISKMAP to run each time DR DOS returns you to the command prompt. If you've already assigned a PROMPT command as part of your AUTOEXEC.BAT file, simply add $X to the end of your existing PROMPT string.

I also use this technique to run the DR DOS MEM command, which gives me a detailed display of my PC's memory status each time I return to the DR DOS prompt after running an application. The commands to do this are:

```
SET PEXEC=C:\DRDOS\MEM /M
PROMPT $P$G$X
```

Adding the MEM command's /M switch provides a graphical display of the current locations of RAM, ROM, and Expanded Memory Specification (EMS) memory in your computer's system memory map.

If you frequently run the DR DOS task switcher, TaskMAX (which I'll describe in detail later), from the command line, you may find this trick especially useful. It provides a quick way to see how much memory is available for starting another application.

Unfragment Your Files With DISKOPT

Another nice DR DOS feature is the DISKOPT utility. Digital Research tossed this feature into its operating system to spare you the expense of buying a specialized program to keep your hard disk in top shape. You'll find nothing like it in MS- or PC-DOS.

DISKOPT rearranges files on your hard drive that have become fragmented as you delete unnecessary files, enlarge existing files, and create new ones. Hard drives are notoriously sloppy as to where they put the data you save. They store portions of it wherever they can find free space on the disk. This fragments your files.

When your hard disk becomes fragmented, your PC must work overtime to collect the bits and pieces of files that have become scattered over the surface of the data platter. DISKOPT gathers the fragments and places them in adjacent disk sectors. Optimizing a badly fragmented hard disk can offer a substantial increase in the apparent speed and efficiency of your computer.

Unless you follow the step-by-step instructions provided in the DR DOS manual, however, you may become confused the first time you use DISKOPT. After you select a drive for optimization, a visual map of the files stored on your disk appears. Pressing Alt-O and selecting the Optimize Disk option from the Optimize menu starts the defragmentation process. Unfortunately, no prompts tell you to do this. If you do what seems natural to you—for example, pressing the Esc key to pull down the Optimize menu—you are summarily returned to the DR DOS prompt.

Also, before you begin optimization, you may want to use DISKOPT's handy Sort option, arranging your files by name, extension, and date or in sequence according to the disk clusters in which they are stored. Sorting your files before you begin optimizing your disk can boost the performance of applications that routinely sort directory entries before presenting them on the video display.

How to Automate Task Lists

Among the new features available in DR DOS 6.0 is TaskMAX, a task switcher that lets you manage as many as 20 applications in memory. Unlike DOS 5.0, which lets you launch multiple tasks only from the protective custody of the DOS shell, TaskMAX also lets you manage tasks from the DR DOS command line. After you begin to use TaskMAX on a daily basis, you'll wonder how you ever got along without it. TaskMAX even lets you run Microsoft Windows as a task.

TaskMAX makes flipping back and forth between applications effortless. The one thing TaskMAX won't do for you, however, is keep a list of the tasks you use daily and then load that task list each time you start your computer. By default, TaskMAX requires you load and run each application you want to become a part of your active task list each time you start your computer.

Fortunately, DR DOS provides a way, using the /C

TIP C THE TREE. C THE TREE GROW.

After you work with DOS for some time, the branches of your subdirectory system can become snarled. DOS 4.0 introduced the TREE command to help you visualize an otherwise cluttered directory structure.

Because TREE is a so-called external command, you must move to the correct directory or provide a complete pathname (either when you invoke the command or by mentioning the directory in your AUTOEXEC.BAT's PATH statement). Assuming you are in the DOS directory in which TREE.COM resides, you can type:

```
TREE C:\
```

to get a neatly organized picture of the directories on drive C.

With DOS 4.01 or higher, you can add an /F switch to the end of the command to also display the names of each file in those directories:

```
TREE C:\ /F
```

But, if you have more than handful of files, the top branches of your tree will scroll off the screen before you can get a good look. Adding a pipe symbol (|) followed by the MORE command remedies the problem:

```
TREE C:\ /F | MORE
```

If you need to really study the situation, you can use a greater-than (>) redirection symbol to output the tree to your printer:

```
TREE C:\ /F > PRN
```

—G.S.

command switch, to add and remove tasks from your current task list. After you load TaskMAX into memory, you can specify each task you want as part of your task list by repeatedly calling TaskMAX with the /C switch. And you can specify the names of your tasks as individual commands in your AUTOEXEC.BAT file. Thereafter, each time you turn on the power switch, DR DOS will load the desired applications and automatically build your task list.

To build an automated task list, add the following lines to your AUTOEXEC.BAT file, substituting the names of your own applications in the third through sixth lines:

```
SHARE /L:80
TASKMAX /E=4096 /L=1024
TASKMAX /C C:\PW\PW
TASKMAX /C C:\LLPRO\LLPRO
TASKMAX /C C:\CEYES\HEYES
TASKMAX /C C:\CSERVE\CIM
VIEWMAX
```

The first line in the example file loads SHARE.EXE. This DR DOS file-locking and -sharing utility must be loaded before you attempt to load TaskMAX. The next line loads TaskMAX into memory using the /E and /L switches. The /E switch tells TaskMAX the maximum amount of your computer's extended memory that it may use, and the /L switch establishes the maximum amount of EMS 4.0 memory each task may use. (If your computer isn't equipped with expanded or extended memory, you can omit these switches.) The next line loads TaskMAX using the /C switch. Notice that I've specified the full pathname for each application I intend to add to my task list.

Finally, if you prefer to run TaskMAX from within the ViewMAX shell, DR DOS's version of the DOS shell, you can tell DR DOS to load and run ViewMAX after DR DOS builds your task list. In this case, you'll need to perform a few additional steps before using the modified AUTOEXEC.BAT file.

Load ViewMAX, then select the TaskMAX Preferences option from the Options menu. Activate the option labeled ViewMAX Controls TaskMAX? by clicking inside the adjacent check box. Select the Preferences option from the Options menu, then activate the Save Configuration on Exit option. Finally, exit ViewMAX to save the current ViewMAX Preferences file to disk. By selecting these options, you give ViewMAX operational control of TaskMAX each time you start your computer.

Creating Intelligent Configurations

Another contribution DR DOS makes to the PC operating system environment is its expanded batch file commands, including GOSUB...RETURN and SWITCH. Digital Research also throws in a new configuration command: the question mark (?).

This unique command is used only in your configuration file and offers you options for configuring your computer. The ? command is a way to ask for user response in a setup routine.

DR DOS's ? lets you create statements in your CONFIG.SYS file that accept yes-or-no answers. For example, you might need a mouse driver for one computing session but want to eliminate it in another. You also can specify that a memory driver be loaded. If you add the following line to your CONFIG.SYS file:

```
?DEVICE=C:\DRDOS\EMM386.SYS
```

DR DOS will interrupt execution of your CONFIG.SYS file to ask:

```
DEVICE=C:\DRDOS\EMM386.SYS (Y/N)?
```

DR DOS will load and execute EMM386.SYS when you type *Y*. If you type *N*, however, the device statement will be ignored. For clarity, you can add a text prompt (set off by quotations) between the ? and the start of the actual command. Using the following syntax can make the request for information clearer:

```
?"Load EMM386.SYS this session?"
DEVICE=C:\DRDOS\EMM386.SYS.
```

because it prompts the users with a question that demands a response:

```
Load EMM386.SYS this session?
```

You can use the ? with any valid CONFIG.SYS statement. The maximum length

TIP MAKE YOUR KEYBOARD FLY

By default, DOS uses a keyboard repeat rate of 20 repetitions per second; that is, a depressed key will generate 20 characters (or other actions, such as a right-arrow movement) each second. Also, DOS normally waits a half second before repeating a character. Inserting the following MODE command in your AUTOEXEC.BAT file increases the keyboard repeat rate to 30 repetitions per second and lowers the initial waiting period to one-quarter second:

```
MODE CON RATE=30 DELAY=1
```

—J.N.

Program Listing. A sample CONFIG.SYS file containing three subroutines, labeled :WINDOWS, :DATABASE, and :WORDS, this program illustrates one of the possible uses of SWITCH.

```
TIMEOUT = 10
ECHO = Select a Configuration
ECHO =  1 - Microsoft Windows
ECHO =  2 - Database Manager
ECHO =  3 - Word Processor
SWITCH WINDOWS,DATABASE,WORDS
ECHO = Configuration is complete.
EXIT
:WINDOWS
HIDEVICE DEVICE=C:\WINDOWS\MOUSE.SYS
HIDEVICE=C:\DRDOS\ANSI.SYS
RETURN
:DATABASE
HIDEVICE=C:\DBASE\DBMS.SYS
HIDEVICE=C:\DRDOS\ANSI.SYS
RETURN
:WORDS
HIDEVICE=C:\WORDS\VGAWORD.SYS
HIDEVICE=C:\DRDOS\ANSI.SYS
RETURN
```

End

for a line is 128 characters (including the question mark).

Timeout for Your Batch Files

Using the ? in a CONFIG.SYS statement is fine for those times when you're sitting at the keyboard. But what if you leave your computer running when you're not around—perhaps because you operate a BBS—and the power fails? If you want the system to restart automatically without someone having to press a particular key, you'll need to add a TIMEOUT statement, another unique DR DOS command, to your CONFIG.SYS file.

The TIMEOUT command provides a way to monitor the response period for answering a question prompt. It lets you specify the time (in seconds) that DR DOS will wait for you to respond by typing a key. Simply add TIMEOUT before the line that contains the prompt you wish to monitor. For example, the following two lines:

```
TIMEOUT = 10
?"Load EMM386.SYS this session?"
  DEVICE=C:\DRDOS\EMM386.SYS.
```

tell DR DOS to wait 10 seconds after displaying the "Load EMM386.SYS in this session?" message. When the timeout period expires, DR DOS will resume execution of your CONFIG.SYS file.

A better way to allow for interaction between you and CONFIG.SYS is coupling TIMEOUT with DR DOS's batch file command, SWITCH. SWITCH, which must be accompanied by the RETURN command, lets you move between different subroutines within the same CONFIG.SYS file. The sample Program Listing shows how you might put SWITCH to work. If the user doesn't type a 1, 2, or 3 within 10 seconds, program execution resumes at the line:

```
ECHO = Configuration is complete.
```

The SWITCH command causes DR DOS to prompt you to select a configuration subroutine by pressing a number key between 1 and 9. Execution of your CONFIG.SYS file then continues with the selected routine. DR DOS relates the labels (such as :WINDOWS) to the keyboard number keys 1 through 9. The first label is linked to the 1 key, the next label is linked to the 2 key, and so forth. If no operator is present to select a response key within the allotted time, DR DOS defaults to the first subroutine mentioned in the SWITCH command.

That's just a sample of DR DOS and a look at a few of its advantages. In using the operating system, I have come across no compatibility problems or performance problems. Even running Windows from the DR DOS prompt proceeds without a hitch.

One gripe may be worth noting, however: all the publicity that Digital Research (and now Novell) has received for including AddStor Inc.'s SuperStor compression utility in its new operating system. While I have some argument with how SuperStor—or any compression utility—does its job, my complaint goes beyond technical concerns. Contrary to what a number of reviews have implied, you won't find the full complement of Super Stor 2.0 utilities in the DR DOS box. To get AddStor's compression program for floppy disks, for example, you'll have to pay extra. ■

John Wolfskill is a free-lance writer and president of Powertrain Development, a New Hampshire-based software-development company.

PRODUCT INFORMATION

DR DOS 6.0, $99
Novell Inc.
70 Garden Court
Monterey, CA 93940
800-274-4374

DR DOS upgrade to SuperStor 2.0, $39
AddStor Inc.
1040 Marsh Rd., Suite 100
Menlo Park, CA 94025
800-732-3133
415-688-0470

More Disk Space For Free

File compressors can squeeze up to 70 percent more data onto your hard drive and, maybe, save you the cost of an upgrade.

by Patrick Marshall

Hard disk space isn't the vast open plain it once was. A few years ago, a 40MB drive had room enough for DOS, your word processor, a spreadsheet program, all your data files, and still you couldn't see a fence. Today, nearly every software package demands megabytes of territory. Move to Windows with only 40MB, and you'll soon find yourself staring into the barbed wire. The graphical operating system from Microsoft carves a 10MB hole in your drive and Windows applications—Word, for example—can chew up another 8MB.

The obvious solutions are to move files to floppies or buy a larger drive. But there's another alternative: file compression. With the right program, you easily—and sometimes automatically—can squeeze data into much smaller files.

How much smaller? Text files can shrink by more than 50 percent and spreadsheet files by more than 70 percent. Even programs, which are tightly packed to begin with, can be reduced by 25 percent with the right software.

The sidebar "A Real-World Example" gives you an idea of what to expect. Thirteen files—nine text and four spreadsheet—initially took up over 472K of disk space. After compression with a utility called PKZIP, the same files are reduced to only 175K. This leaves almost 300K of space available for new files.

The two main types of data crunching are file compression and disk compression. A file compression program squeezes and extracts only files you've tagged. A disk compression program works in the background, automatically decompressing files as you call on them and compressing them when you're finished.

Popular file compression programs include LHA (available free on bulletin boards), PKWARE's PKZIP, and System Enhancement Associates' ARC+ PLUS. Disk compression programs include AddStor's SuperStor and Stac Electronics' Stacker.

How File Crunchers Work

File compression programs were created years ago by the bulletin board crowd to save computer time and long-distance dollars while uploading and downloading files (see the sidebar "Uploading Files? Save Money!").

Most compression programs work on the same principle: They eliminate redundancy. Specifically, they use a mathematical algorithm—the Lempel-Ziv algorithm, first published by two Israeli mathematicians in 1977—to find patterns in a file's data. Where repetitive strings occur, the compression program inserts a token, which takes up much less space.

For example, "How many pickled peppers did Peter Piper pick? Peter Piper picked a peck of pickled peppers" could be shortened to "How many *l+ #did @ ! *? @ ! *+ a peck of *l+ #." Even after adding an index (which is created by the routine and tells the program where to put the pieces when it decompresses the file), a compression program can crunch the average text file to slightly less than half its original size. (Refer to the sidebar "Tricking DOS to Save Megabytes" to see how compression programs also achieve space savings by repacking the hard drive.)

This technique is even more effective with spreadsheet and database files—compression rates above 70 percent are common—because they usually contain more repetitive data. Executable files are the least compressible. (It's impractical to compress them anyway, if you have to decompress them manually before you can use them.) But some programs—PKWARE's PK-Lite is one—compress EXE and COM files by about 25 percent, yet leave them in an executable state.

The Big Trade-Off: Speed Vs. Space

Even though all the major file compression programs are based on the same algorithm, they do differ. Every application implements the algorithm in a different way, achieving different combinations of the two trade-offs of compression: speed and degree of compression. (Refer to the sidebar "Cutting Image Files Down to Size" to find out more about trade-offs in the compression of image files.)

PKZIP, for instance, usually wins the speed contests, both in compressing and extracting files. It's no coincidence, however, that the fastest algorithm also is one of the least efficient. If file size, not speed, is your principal concern, you'll want to look at LHA and Robert K. Jung's ARJ. Keep in mind, however, that these are only generalizations: In a test, PKZIP not only managed the fastest speeds on my single sample text file, it also placed first in degree of compression (see the benchmark chart on page 59).

All of the leading file compression utilities have been refined to the point where they're highly reliable. Corruption of files—especially of nongraphics files—is rare. Remember, though, that the file formats of file compression programs are mutually incompatible, so if you're sending a file to someone else, that person must have the same file compression program (and often the same version) to extract the file.

As usual, there are exceptions to every rule. To begin with, three programs—ARC+ PLUS, LHA, and PKZIP—let you create self-executing files; that is, when you run the crunched file, it automatically uncrunches itself. Also, programs that convert one compression format to another are starting to appear on the market. Loon Valley Software's D'Compress for Windows, for example, converts PKZIP, ARC, and LHA files to the ARJ format.

Although mutual incompatibility remains a problem, the good news is that compression programs are affordable enough for most users to own several. LHA, for instance, can be downloaded at no charge from a number of bulletin boards. As shareware, PKZIP also can be downloaded but, if after using it, you decide to keep the program, you'll owe PKWARE $47 (see the product information box on page 60).

Indeed, file compressors are getting almost as accessible as they are inexpensive. The factor most responsible for keeping these programs from becoming more popular is that they haven't been easy to

A REAL-WORLD EXAMPLE

The following text and spreadsheet files were compressed with PKZIP, a popular file-crunching utility from PKWARE. The utility shrank the text files by 19 percent to 68 percent. It shrank all the spreadsheet files by over 70 percent. On average, it reduced the files by 63 percent, reclaiming almost 300MB of disk space.

The results vary widely among the text files, because PKZIP is more effective when a file contains a higher number of redundant character strings; some files had a much higher ratio of unique strings. In general, spreadsheet and database files compress better, because they have more repetitive data than text files do.

—P.M.

THE RESULTS

	Before	After	Shrunk by:
MUSIC.TXT	50,100	40,360	19.44%
IQUEST.TXT	19,865	8,856	55.42%
IQUEST2.TXT	15,497	5,271	65.99%
DOCS.TXT	23,176	7,494	67.66%
ARTICLE.TXT	2,099	975	53.55%
CIS.TXT	77,650	31,375	59.59%
COLLEGES.TXT	17,366	7,823	54.95%
AUDIO.TXT	49,491	15,738	68.20%
URBAN.TXT	1,437	1,009	29.78%
DISK.WQ1	107,314	28,138	73.78%
BUDPLAN.WQ1	19,956	4,790	76.00%
LOANS.WQ1	51,336	12,242	76.15%
EXPENSES.WQ1	37,158	9,735	73.80%
TOTALS	472,445	173,806	63.21%

UPLOADING FILES? SAVE MONEY!

Because it reduces upload time, compression can save you big bucks if you occasionally transfer files via a private BBS. For example, suppose you're sending a 200K file from Boston to San Francisco. At 2400 bps, this takes about 30 minutes and will cost you about $7.50 in telephone charges. If you first crunch that file to 100K, it takes only about 15 minutes and costs you a little over $3.75. You'll soon save enough to more than cover the price of the compression program.

—P.M.

use: Most lack an interface and require users to enter potentially complex arguments at the naked DOS prompt.

Fortunately, some file compression programs now include shells with menus for inexperienced users. ARC+ PLUS, for example, comes with a built-in menu interface; PKWARE sells an optional shell, called PKZIP Menu; and a handful of third-party shell applications are available.

What's more, many file management programs—both in DOS and Windows—are starting to integrate one or another of the compression methods. Lotus's Magellan first offered built-in file compression using PKZIP back in 1990. And Executive Systems' XTree Gold added the same PKZIP feature to its strong set of file management tools approximately a year ago.

The Background Work Of Disk Compressors

The biggest drawback to file compression programs is that they entail a good deal of work by the user. Even with a good menuing interface, the task of tagging files for compressing and locating them to perform decompression before use can be a deterrent. This is why disk compression programs, such as AddStor's SuperStor and Stac Electronics' Stacker, have grown popular.

A disk compression program uses the same type of algorithm as a file compression program. The crucial difference is that it works in the background rather than requiring you to select files.

As with file compressors, disk compressors turn in a surprisingly wide range of performances. One program may take as few as 12 seconds to decompress a 3.5MB file, while another may take more than 40 seconds. Compression ratios can vary a lot, too—from about 2:1 to 3:1 for a text file. In short, by adding a disk compression program, you can assume that you'll nearly double the space available on your hard drive, but you'll have to decide whether you'd rather squeeze out a

CUTTING IMAGE FILES DOWN TO SIZE

Where compression is concerned, image files are a special case. And if you've started saving image files—especially color image files—for use in newsletters, screen blankers, and the like, you know how quickly they can take over your hard drive. A single, full-color 3- by 5-inch scanned image can require more than 3.5MB of storage space.

The problem is, while the traditional Lempel-Ziv-based algorithm can eliminate redundancy even in high-resolution gray-scale images, there is very little redundancy to remove when it comes to the pixels of a full-color image: Nearly every pixel is likely to be a slightly different color from its neighbor. As a result, using a program such as PKZIP on a full-color TIF file almost never results in a size reduction of more than 10 percent, and the compression often is less.

The solution is several types of "lossy" compression algorithms—that is, compression in which information is lost. Lossy compression works by eliminating unneeded information from image files.

The fact is, the human eye can't perceive the full 16.7 million different colors that can be displayed in a 24-bit image. And, because the eye fails to distinguish between very slight color differences in neighboring pixels, a compression program can treat nearly identical colors as identical, allowing compression without perceptible loss of quality. To achieve greater compression, the program simply recognizes a smaller range of colors. When compression of color images is handled this way, you can obtain compression ratios as high as 40:1. At ratios of 10:1 or less, the differences are very hard to perceive.

Several types of lossy compression exist, though one method—JPEG, named after the Joint Photographic Experts Group that developed it—is emerging rapidly as the industry standard.

For the most part, JPEG compression is available only through hardware-assisted packages. Some high-end, full-color video boards, such as Opta's MonaLisa, include a JPEG plug-in module to compress image files quickly and extract them when they are called upon as an option. But JPEG also is appearing rapidly as a software-only compression option in image-editing applications, such as Aldus's PhotoStyler and Computer Associates' Cricket Image.

—P.M.

little more space or speed up the compression process.

Disk Compression Pitfalls

Even with the best disk compressors, often you'll notice at least a small degradation in your system's overall performance, particularly when large files are accessed. To speed up the processing, many of the programs—Stacker AT/16 ($249) and Cyrix's Expanz! ($99) are examples—employ hardware help in the form of a coprocessor on an 8- or 16-bit board. But unless your files are large, you'll find that the hardware-assisted programs are not all that much faster than their software-only counterparts.

Unfortunately, a slight loss of speed is not the only potential problem presented by disk compression programs. The most serious threat is the danger of having another memory-resident program running, which could mean the loss of significant amounts of data if a conflict occurs. After all, if a file compression program corrupts a file, you may lose most of that one file. But if a disk compression program crashes—and, being RAM-resident, that is always a possibility—it may mean the loss of a drive full of data. Though the risk is small, the potentially catastrophic consequences continue to cause many users to shun disk compression programs.

And other, less drastic problems can come up when opting for a disk compressor. Upon installation of some disk compression programs, for instance, you must repartition your drive, a task that can be nerve-wracking for uncertain novices.

The main reason you have to partition your drive is that disk compression programs require you to leave at least a portion of your drive uncompressed; the computer has to boot from uncompressed files before the compression program can load into memory. And if you use Windows, the uncompressed portion of your drive may have to be quite large.

These programs can't compress Windows swap files (files Windows uses to store data on your hard disk temporarily and which usually occupy about 4MB of memory on a system with between 2MB and 8MB of RAM). Also, caching programs that use a write cache—Windows' SmartDrive is among them—don't work with most disk compression programs. If you use a caching program, you must disable the write cache if possible or refrain from using it with the compressed drive.

Some applications—again, Windows is among them—aren't easy to install on compressed drives, because such drives require that you disable all memory-resident programs. This means that you must save enough room on your uncompressed drive to install those applications and then copy the files to the compressed drive. Not only does that make installation a more complicated process, but in the case of installing Windows, it means keeping at least 10MB in unused reserve on the uncompressed drive.

Another potential headache is that most of the disk compression programs ask you to estimate how much compression you're going to achieve (the programs need to reserve an appropriate number of "pointers" to data within the allocated space). If you estimate a high ratio, say 10:1, the program reserves more pointers. But pointers take up about 2 bytes each. So make sure not to overestimate by too much. On the other hand, underestimating is even worse: If you run out of pointers, the program tells you the disk is full, even when it isn't.

TRICKING DOS TO SAVE MEGABYTES

Not all space savings can be credited strictly to the algorithms of disk compression programs. Disk compressors not only shrink files, they repack them on the drive, just as disk optimizers do. And, by playing a little trick on DOS, disk compressors do a much better job of packing files than DOS itself.

DOS is a profligate waster of disk space. The smallest piece of real estate DOS deals with when laying down files is the cluster, which is composed of 4, 8, or 16 512-byte sectors, depending on the type and size of the hard drive. If a file holds only 2 bytes of information, it still takes up an entire cluster, because DOS reads the cluster as occupied.

Similarly, if the file entirely fills one cluster and spills over to occupy 4 bytes of the next cluster, DOS reads the file as occupying two full clusters. This means you easily could have DOS telling you that your 40MB hard drive is full when as much as 10MB or more of the sectors are unused.

Disk compression programs reclaim that unused space by getting between DOS and the disk and letting DOS think that clusters are not shared, when in fact they are. With disk compressors, the sector, and not the cluster, works as the smallest storage space.

—P.M.

BENCHMARK TESTS*

	Compression Time (seconds)/Size	Extraction Time (seconds)
PKZIP	2.8/33K	0.8
LHA	4.1/35K	1.2
ARC+ PLUS	5.8/37K	2.6
ARJ	3.9/33K	1.4

*** All tests were performed on a 25MHz 386 using an 85K text file.**

PRODUCT INFORMATION

FILE COMPRESSION PROGRAMS:

ARC+ PLUS 7.12, $89.95
System Enhancement Associates Inc.
21 New St.
Wayne, NJ 07470
201-473-5153

ARJ 2.30, $40
Robert K. Jung
2606 Village Road W.
Norwood, MA 02062
617-769-5999

LHA
Free on bulletin boards

PKZIP 1.1, $47
PKWARE Inc.
9025 N. Deerwood Dr.
Brown Deer, WI 53223
414-354-8699

DISK COMPRESSION PROGRAMS:

DoubleDensity, $79.95
Abacus
5370 52nd St. S.E.
Grand Rapids, MI 49512
800-451-4319
616-698-0330

Expanz! 2.10a, $99
Cyrix Corp.
2703 N. Central Expressway
Richardson, TX 75080
800-447-0250

Stacker 2.0, $149
Stac Electronics
5993 Avenida Encinas
Carlsbad, CA 92008
800-522-7822

SuperStor 2.0, $139
AddStor Inc.
1040 Marsh Road, Suite 100
Menlo Park, CA 94025
800-732-3133
415-688-0470

Finally, if you decide you don't want the disk compression program after all, removing it can be a little tricky. Some programs, such as Abacus's DoubleDensity, include a removal option; otherwise, you have to locate the disk compression program's hidden files and delete them. And you also may have to reformat your drive.

Weigh the Pros and Cons

It is true that there are no free lunches: Both data-compression methods discussed here result in at least a small loss in performance. On top of that, file compressors also involve no small amount of file housekeeping. Full disk compression programs eliminate the housekeeping chores because they work transparently in the background, but they, too, impose their own inconveniences and present some risks.

Although disk compression programs are now quite reliable, it's the qualification implied in the word *quite* that makes me nervous. Maybe I'm being silly, but I can't bring myself to trust my entire drive to a RAM-resident program of any sort. I'm willing to put up with the housekeeping headaches of archiving with a file manager, rather than entrust my data to a disk compression program that sits precariously in memory.

One thing is clear, however: Data compression is a much cheaper way to expand your available storage than buying a larger hard drive. And you'll continually save time and money when transmitting files if you compress them prior to sending. ■

Patrick Marshall is a contributing editor for InfoWorld *magazine.*

TIP HELPFUL ASSOCIATIONS

A most useful feature of DOS 5.0's shell is its ability to associate data files with the programs used to create them. Setting up an association causes DOS to run the appropriate application program automatically when you click on a data file's name.

The following steps, for example, would alert DOS to launch Microsoft Word and the selected document each time you click on a filename ending with a DOC or WP5 extension:

```
1. Highlight WP.EXE in the file listing.
2. Select the File menu.
3. Select the Associate option.
4. Type:

     DOC WP5

   in the Extensions box.
5. Select OK.
```

To use this same shortcut with other applications, simply substitute the appropriate file extension in step 4, above.

—J.N.

The Importance Of Good Thinking

Sound logic and advance preparation are as essential to a well-written and -designed program as the code itself.

by John Wolfskill

Just because you're interested in DOS doesn't mean you have to take up programming. But it might help. No matter how good DOS is and how many reasonably priced programs are available for it, there's always a niggling problem in need of a solution. In my case, it was a new printer with fancy options that I just couldn't take advantage of from within my word processor. That's no longer an obstacle since I wrote a QBasic program, PPRIMER.BAS, that lets me tap my printer's full potential from the DOS prompt.

You can adapt my program to work with the printer of your choice, but printing isn't this program's only function. PPRIMER.BAS is a teaching tool that can show you how to break into programming and break down computing barriers.

In the initial installment of this series on QBasic programming, I discussed the four building blocks that make up a typical computer program: initialization, input, process, and output code. This time around, I cover a topic even more integral to the process: thought—the cogitations and preparations essential to producing a satisfactory final product.

Assessing the Problem

Before you can write a successful computer program, you must define the problem it will solve. I usually find a comfortable chair, then sit back with pad and pencil in hand to think. I write down everything that comes to mind. Although this may seem like an excuse to goldbrick, it's an important step. By airing your ideas, you usually can weed out those that may be impractical or difficult to implement. My first question always is: "Why should I write this program?"

In the case of my printer difficulty, I summarized the situation as follows:

```
A. Problem = Difficult to set up printer from
   DOS prompt.
   1. Want to take advantage of special features.
   2. Want to set letter-quality mode, fonts,
      and so on before using DOS's TYPE and
      PRINT commands.
   3. Need a way to set up printer for use with
      applications that don't support its special
      features.
```

Editor's Note: *In* DOS Resource Guide *Number 5 (page 58), we printed PPRIMER.BAS in its entirety, but limited space precludes our publishing it again here. If you don't have this issue, you may order it (see the back cover for details) or send us a formatted $5\,^{1}/_{4}$- or $3\,^{1}/_{2}$-inch PC disk in a self-addressed, postage-paid mailer. We will copy PPRIMER.BAS onto the disk and return it to you as soon as humanly possible. Address requests to PPRIMER.BAS,* DOS Resource Guide, *80 Elm St., Peterborough, NH 03458.*

Once your problem is defined, the next step is making a list of alternatives to writing the program. Don't overlook this part of the creative process. In the past, I've devoted a considerable amount of time and effort to writing a utility, only to discover that someone already had written a better one!

Here are the choices I considered before writing a single line of PPRIMER.BAS:

```
B. Alternatives
   1. Load word processor or text editor for
      simple tasks.
   2. Purchase a utility (does one exist?).
      a. Check available sources (magazines,
         software catalogs, bulletin board
         systems) to see if a utility already is
         available.
```

In the end, though, none of the alternatives panned out, because I couldn't find a utility that had *all* the features I felt I needed.

Defining Program Requirements

With the problem both defined and justified, the next step is determining the features your program should provide for the user. My initial list usually includes the major functions I want the program to perform, details about aesthetics, and comments about how to approach the actual coding. Because the list will become the "skeleton" of my program, I try to include as many details as possible. I uncovered plenty of must-haves for PPRIMER.BAS:

```
C. Requirements
   1. Must work with all printers.
   2. Must be quick and easy to use.
   3. Must provide an attractive user interface.
   4. Must be able to save control strings for
      future use.
      a. A maximum of 12 control strings will
         suffice.
      b. A small data file may be the best
         approach for saving the data.
   5. Each control string should be identified by
      a plain-English phrase of at least 30–40
      characters.
   6. For simplicity, the program also should
      provide a way to edit the text and data
      portions of the string separately.
   7. Two or three keystrokes should be sufficient
      for sending a control string to the printer.
   8. Should include a way to perform a printed
      test to verify the string has arrived at the
      printer.
   9. The program also should provide audible
      feedback to the user.
```

The next step is one that beginning programmers sometimes overlook: deciding the intended audience for the program. If you're writing a utility for your own use, you may decide to forgo the extra code that lets the program present a "pretty face" to the user. Because you're anxious to see your program in action, you also may decide to skip other niceties, such as error-trapping or help screens. This approach to program design generally is frowned upon by professional programmers. However, within reason, it may be acceptable for the programs you write for your own use. Be careful, though. If your program is too cryptic, even *you* may not remember how your program operates six months from now. (For advice on how to create an interface, see the sidebar, "Designing a User Interface," on page 64.)

Go With the Flow

With your basic outline complete, it's time to pick up a pencil and map out a rough sketch of your program's design. The time-honored tool for doing this is a logic flowchart, a blueprint that can help you identify inconsistencies in your program's organization and logic. A flowchart also can help you define major sections of program code and explore how these sections will fit together eventually. Among the most valuable functions of a flowchart is its ability to show you exactly how your program will operate, even before you write the first byte of code.

To introduce you to the discipline of flowcharting, I'll walk you step by step through the logic behind my design for PPRIMER.BAS (see the figure, "PPRIMER.BAS in Profile"). As I describe the major sections of the program code, you may find it helpful to follow along, using the source listing as a guide. Don't worry if you don't understand all of the program's details. I'll discuss the specifics of how each program routine works at another time. The program's top block represents its initialization phase (sections A–J). This block sets up the error-handling routine, prepares the video display for operation, displays the input screen, and loads the disk file that contains data for the printer control strings. If no errors occur during initialization, the program enters the main processing loop (section K), ready for action.

Each rectangular block in the flowchart represents a major process the program must perform. The diamond-shaped blocks indicate points at which the program must make a decision. If you recall, a process can be defined as a section of program code that calculates a result, usually based on input from the user.

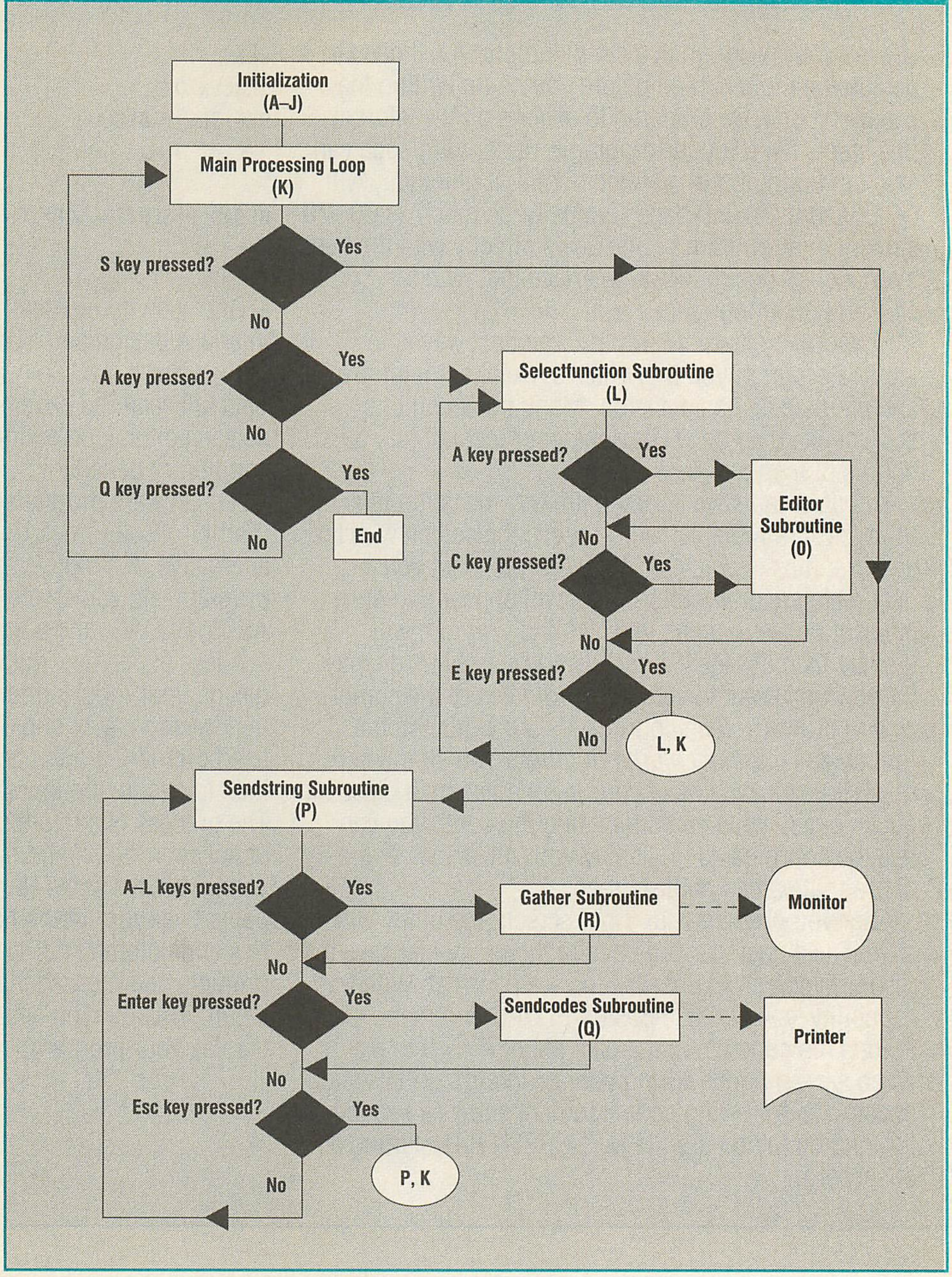
To identify the major processes required for PPRIMER.BAS and to determine the order in which to perform them, I used my "wish list" of proposed features. After careful study, I decided that in addition to the initialization and main processing sections, I needed one process to assign and edit printer control strings (the Selectfunction subroutine) and another to send them to the printer (the Sendstring subroutine). A few other subroutines round out the program.

So far, I've defined three major processes: the main loop, the Sendstring subroutine, and the Selectfunction subroutine. Let's follow the divergent paths to see where each leads.

After initialization, the main processing loop takes over and prompts the user to press a key. When the user types in a response, the main processing loop tests for the S, A, and Q keys in turn. If the user presses the S key, the Sendstring subroutine (section P) takes over. If not, the main processing section then checks to see if an A key was pressed. If so, control shifts to the Selectfunction subroutine (section L). When the Selectfunction or Sendstring function completes its job, program control automatically returns to the main processing loop (K). If the Q key is pressed, the program ends. If the user types a letter other than *S* (for *send*), *A* (for *assign*), or *Q* (for *quit*), program control returns to the top of the loop, and PPRIMER.BAS awaits further user input. At every decision point, the program must provide one course of action if the condition is met and another if it is not.

If you were a first-time user of PPRIMER.BAS, you most likely would choose option A from within the main loop. The program then would transfer control to the Selectfunction subroutine, which provides a means of adding or editing as many as 12 printer control codes. Like the main processing section, this subroutine prompts you to make a choice—by pressing *A* (for *action*) to create/edit the text describing a control string, *C* (for *control*) to create/edit a control string, or *E* to exit the routine.

PPRIMER.BAS IN PROFILE



To permit separate editing of both the text describing a control string's action and the control-code portions of the string, PPRIMER.BAS needs yet another process. If you press the A or C key, the program calls the Editor subroutine (section O). Within this subroutine are other minor processes (not indicated in the flowchart) that handle the job of editing or adding new control strings to the list. When the user presses *E*, control returns to the Selectfunction subroutine, which in turn immediately passes control to the main processing loop.

If you then want to test your newly created control strings, press the S key (from the main processing loop) to proceed to the second major branch of the program, the Sendstring function. Pressing a letter from *A* through *L* at this point invokes the Gather subroutine (section R). Gather displays on your monitor the control string corresponding to the letter you pressed, and then it returns program control to the Sendstring function. If you press the Enter key after selecting a string to send, PPRIMER.BAS calls Sendcodes (section Q), the routine handling the actual work of sending the control code

DESIGNING A USER INTERFACE

Unless your program is a simple utility, you'll need to design the input screen before you begin writing the program code. Be prepared to expend a little effort at this stage. Even if your program's underlying code can leap tall buildings in a single bound, it ultimately will be judged by its interface. Typically, as much as 60–70 percent of a program's code deals directly with the user interface, so designing an interface that works efficiently is important.

How do you learn the ropes? The best way is to study a program you think has an excellent interface and emulate its basic design. Many books on this subject exist, so don't overlook your local library as a source of inspiration.

You also can save yourself time by investing in a utility program that lets you "paint" the screen with text and graphics characters. Such programs let you designate areas where prompts will appear and draw colorful boxes in areas in which the program will display its messages. Typically, these utilities display the row and column screen coordinates of the cursor as you draw. (Screen coordinates are written in the format *row number,column number.*) You simply jot down the coordinates of your input areas on a piece of paper. Later, while you are writing the code, you can use the coordinates to display your messages in the appropriate areas of the screen.

After you prepare your input screen, the utility will save it as a data file. But be sure the utility can save the resulting screen in Basic's binary file format. With this capability, you can use Basic's BLOAD command later to load the contents of the data file directly into the video screen buffer during your program's initialization phase. The following code fragment shows a sequence for loading the binary file MYSCREEN.BAT starting at offset zero:

```
CLS                      'clear the screen
SCREEN 0                 'shift to text mode
DEF SEG = &HB800         'point Basic's data segment
                         '(DS) to the start of
                         'the text screen buffer
BLOAD"MYSCREEN.SCR",0    'load the file starting at
                         'offset zero
```

One such commercial product that automates interface design and is compatible with QBasic is Hi-Screen Pro II 2.0 (Softway Inc., 415-896-0708, $395). This full-featured development system includes a screen editor, icon editor, menu editor, graph editor, and library generator.

A shareware program named Laughing Dog (Yardbird Software, 818-704-6402, $35) is another alternative. A simple-to-use screen designer, this program lets you create text screens and save them in ASCII, BLOAD, and a number of other formats.

A less expensive, but more time-consuming, way to design your input screen is with a pencil, a piece of thin tracing paper, and a large sheet of graph paper. (I used ordinary graph paper with eight squares to the inch to design the input screen for PPRIMER.BAS.) The process is straightforward. Using a marker, you simply cordon off and label a rectangle 25 blocks high and 80 blocks wide. Next, place a sheet of tracing paper over your graph paper "screen." Finally, mark the row and column coordinates where you want your prompts and messages to appear. Later, you'll have all the information you need to tell Basic exactly where to display your program's messages and prompts.

—J.W.

string to the printer. If you press the Esc key, program control returns to the main processing loop.

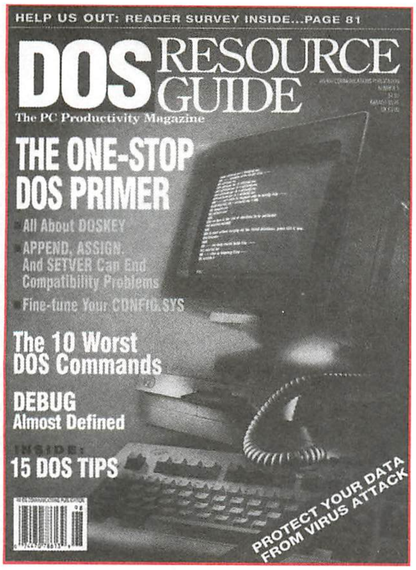
As you can see, logically ordering a program's processes—even with a relatively short program, such as PPRIMER.BAS—can prove challenging. A flowchart is an invaluable tool for defining the major processes and helping to visualize how the program will work. Later, as you write the program code, it will serve as a guide to keep you on track. Although you may add more routines than appear in the original flowchart, you'll know that your logic is sound.

Ready, Set, Code

Now that the advance preparations have been completed, you're ready to start coding. And that's where I'll pick up next time, when my discussion will center on the inner workings of PPRIMER.BAS's various routines. In the meantime, you can get a head start—by looking through the program's listing to familiarize yourself with the lingo (a peek at a QBasic user's manual also wouldn't hurt) and perhaps by entering all, or at least a few, of the routines. You'd be surprised at how easily your brain can absorb the structures that your fingers are typing. ■

John Wolfskill is a free-lance writer and president of Powertrain Development, a New Hampshire-based software development company.

SUBSCRIBE TODAY!



HELP US OUT: READER SURVEY INSIDE...PAGE 81
DOS RESOURCE GUIDE
The PC Productivity Magazine
THE ONE-STOP DOS PRIMER
All About DOSKEY
APPEND, ASSIGN, And SETVER Can End Compatibility Problems
Fine-tune Your CONFIG.SYS
The 10 Worst DOS Commands
DEBUG Almost Defined
INSIDE: 15 DOS TIPS
PROTECT YOUR DATA FROM VIRUS ATTACK

BUCK THE TREND! STAY WITH DOS!

☐YES! Send me 6 Issues of DOS Resource Guide for $23.70*.

That's a DOLLAR OFF EACH ISSUE on the newsstand!

SUBSCRIBE TODAY AND SAVE 20% OFF THE COVER PRICE

NAME ____________________
ADDRESS ____________________
CITY ____________________ STATE ______ ZIP ______

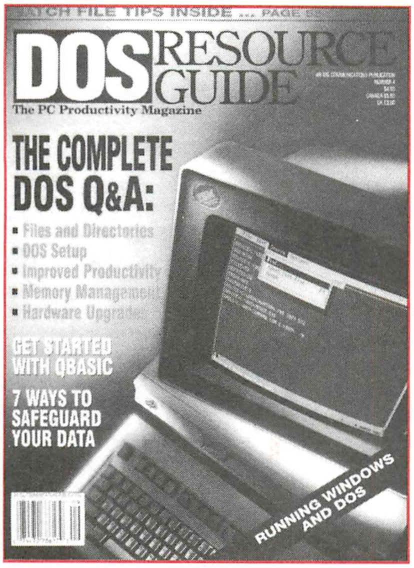
(Check, Money Order, or Credit Card—Payment must accompany your order)

☐ AMEX ☐VISA ☐MASTERCARD ☐DISCOVER

CREDIT CARD # ____________________
EXPIRATION DATE: ____________________
SIGNATURE: ____________________

ID6C1

*All foreign subscriptions must be paid in U.S. funds drawn on a U.S. bank.
*Canada and Mexico: $29.70 (includes GST)
*Maryland residents add 5% sales tax.
*All other foreign (Surface Mail): $41.70
*Foreign Air Mail: $53.70



BUCK THE TREND! STAY WITH DOS!

☐YES! Send me 6 Issues of DOS Resource Guide for $23.70*.

That's a DOLLAR OFF EACH ISSUE on the newsstand!

SUBSCRIBE TODAY AND SAVE 20% OFF THE COVER PRICE

NAME ____________________
ADDRESS ____________________
CITY ____________________ STATE ______ ZIP ______

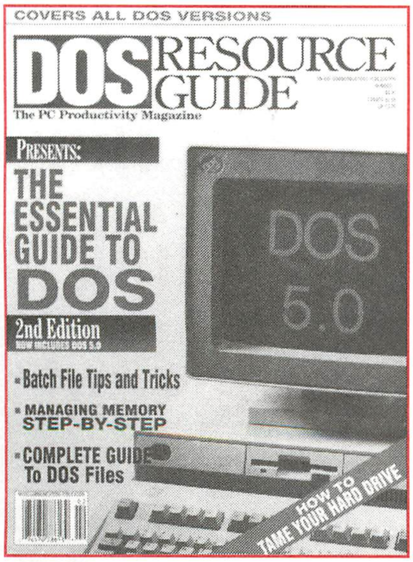
(Check, Money Order, or Credit Card—Payment must accompany your order)

☐ AMEX ☐VISA ☐MASTERCARD ☐DISCOVER

CREDIT CARD # ____________________
EXPIRATION DATE: ____________________
SIGNATURE: ____________________

ID6C2

*All foreign subscriptions must be paid in U.S. funds drawn on a U.S. bank.
*Canada and Mexico: $29.70 (includes GST)
*Maryland residents add 5% sales tax.
*All other foreign (Surface Mail): $41.70
*Foreign Air Mail: $53.70



COVERS ALL DOS VERSIONS
DOS RESOURCE GUIDE
The PC Productivity Magazine
PRESENTS:
THE ESSENTIAL GUIDE TO DOS
2nd Edition
NOW INCLUDES DOS 5.0
DOS 5.0
Batch File Tips and Tricks
MANAGING MEMORY STEP-BY-STEP
COMPLETE GUIDE To DOS Files
HOW TO TAME YOUR HARD DRIVE

BUCK THE TREND! STAY WITH DOS!

☐YES! Send me 6 Issues of DOS Resource Guide for $23.70*.

That's a DOLLAR OFF EACH ISSUE on the newsstand!

SUBSCRIBE TODAY AND SAVE 20% OFF THE COVER PRICE

NAME ____________________
ADDRESS ____________________
CITY ____________________ STATE ______ ZIP ______

(Check, Money Order, or Credit Card—Payment must accompany your order)

☐ AMEX ☐VISA ☐MASTERCARD ☐DISCOVER

CREDIT CARD # ____________________
EXPIRATION DATE: ____________________
SIGNATURE: ____________________

ID6C3

*All foreign subscriptions must be paid in U.S. funds drawn on a U.S. bank.
*Canada and Mexico: $29.70 (includes GST)
*Maryland residents add 5% sales tax.
*All other foreign (Surface Mail): $41.70
*Foreign Air Mail: $53.70

NO POSTAGE
NECESSARY
IF MAILED
IN THE UNITED
STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 73 PETERBOROUGH, NH

POSTAGE WILL BE PAID BY ADDRESSEE

DOS RESOURCE GUIDE
PO Box 802
Peterborough NH 03458-9971

NO POSTAGE
NECESSARY
IF MAILED
IN THE UNITED
STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 73 PETERBOROUGH, NH

POSTAGE WILL BE PAID BY ADDRESSEE

DOS RESOURCE GUIDE
PO Box 802
Peterborough NH 03458-9971

NO POSTAGE
NECESSARY
IF MAILED
IN THE UNITED
STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 73 PETERBOROUGH, NH

POSTAGE WILL BE PAID BY ADDRESSEE

DOS RESOURCE GUIDE
PO Box 802
Peterborough NH 03458-9971

YES!

Send me 6 Issues (one year) of DOS Resource Guide for $23.70.

Save me a buck an issue!

Building A QBasic Database

You get a program you can really use and insight into the workings of DOS 5.0's bundled programming language.

by David Bannon

A program written in QBasic is a bit like a model airplane. Some people are content to admire the lines and colors of a model built by someone else, while others have the drive and the patience to glue the various pieces together themselves.

NAMES.BAS, my QBasic database for names and phone numbers, is one of the easiest "models" to build. You can simply type it in and use it, but I recommend reading on to see how it's glued together. You might find yourself agreeing that programming in QBasic—the successor to GW-Basic that comes free with DOS 5.0—can be satisfying to even the most reluctant users.

Getting Started

NAMES.BAS uses simple commands that you can duplicate and employ in other QBasic programs. It lets you search your list by company name, first, or last name, and includes a simple color menu. NAMES.BAS has few of the extras a commercial database might offer, but it's easy to write and fun to pass along to friends. What's more, after you've mastered the basics, NAMES.BAS lets you throw in your own personalizing frills.

To begin, type `QBasic` at the DOS prompt. When typing NAMES.BAS (exactly as it is shown in Program Listing 1), remember to save often. QBasic automatically checks syntax and even changes lowercase letters to uppercase where needed. And QBasic offers something many other programming environments do not: a comprehensible help system.

Opening the Menu

The first major element in NAMES.BAS is the menu, which is in a SUB-END SUB procedure—that is, a subprogram. You set up the procedure in the second line of the program by typing:

```
DECLARE SUB
```

Then create the SUB procedure by pulling down the Edit option from the standard menu bar at the top of the screen and selecting the NEW SUB entry from the options list. At the blank screen, start typing the lines that begin with the SUB MENU command about two-thirds of the way through the program listing. QBasic then reads the END SUB command and automatically returns to NAMES.BAS. You now can type the rest of the database program.

The Database Design Explained

You can examine NAMES.BAS in three simple steps, each of which you may alter to create new QBasic programs. The first step is defining the variable File$, which names the file in which the database information is stored (count down 19 lines to find this point in the program). NAMES.BAS calls this file NAMES.TXT, but you can rename it to suit your needs. Next, the Form$ variables define the tabular format for the data in

NAMES.TXT. Later, the PRINT USING statement uses this format to print out your list of names. When typing the lines involving the Form$ variable, be sure to get the spacing exactly as shown.

The second step sets up a main menu that consists of three options and a prompt. The DO loop with the INKEY$ function (directly under the comment "Get the user's choice") waits for user input. Unlike many input routines, this one doesn't require the user to press Enter, but rather one of the three defined options: Enter a name, Search for a name, or Quit.

Step three is the meat of the program: SELECT CASE. This function divides the NAMES.BAS database into the three options mentioned above; it begins right after the options in the program listing. Entering 1 executes the first section of SELECT CASE. The NAMES.TXT file defined by File$ is opened for APPEND as file 1—to slightly paraphrase the program code itself. (This means that if File$ does not exist, QBasic creates it; if File$ does exist, QBasic adds—appends—new information to it.)

The program then prompts the user to enter the name, area code, phone number, and so on. If the user types *END* in the last-name field, which the program indicates is the way to complete an entry, the program closes the file and returns to the main menu. WHILE...WEND—a structure that executes a series of statements as long as a certain condition is true—uses QBasic's UCASE$ function to let the user type *END* with any combination of upper- and lowercase characters.

The second section of the SELECT CASE block executes when the user presses 2. Notice the location of the OPEN and CLOSE statements in the WHILE...WEND loop (a loop is a program section that repeats until a certain condition is satisfied). The first search for a database name is accomplished by the DO WHILE (NOT EOF(1)) loop. The easiest way to back up and

Program Listing 1. NAMES.BAS is a simple database for saving names, addresses, and phone numbers. You can alter and extend its data fields to include other types of information.

```
DEFINT A-Z
DECLARE SUB Menu (Row, Column, Option$(), Escape, Selection)

CONST TRUE = -1
CONST FG = 7, BG = 4

DIM Main$(1 TO 10)
FOR X = 1 TO 10
  READ Main$(X)
NEXT


'Display the main menu
MainMenu:
DATA NAMES DATABASE, Keep track, of names and numbers!, MAIN MENU, -
DATA 1. ENTER a name, 2. SEARCH for a name, 3. Quit, -
DATA Select a number

CLS
CALL Menu(6, 30, Main$(), LastKey, Selection)
COLOR 7, 1

'A simple database program for entering names, phone numbers, and
'company names and searching for them.

File$ = "NAMES.TXT"    'Database filename
Form1$ = "\            \\          \"
Form2$ = "(\ \)\        \"
Form3$ = "\                             \"



'Get the user's choice
DO: Choice$ = INKEY$
LOOP UNTIL Choice$ = "1" OR Choice$ = "2" OR Choice$ = "3"

SELECT CASE Choice$
  CASE "1"          'Add a name to the database
    OPEN File$ FOR APPEND AS #1
    WHILE UCASE$(Last$) <> "END"
    CLS : LOCATE 11, 25: PRINT "ADD a name to the database"
    LOCATE 12, 25: PRINT "(To stop, enter END as a last name)"
    LOCATE 14, 25: INPUT "Last name: ", Last$
    IF UCASE$(Last$) <> "END" THEN
      LOCATE 16, 24: INPUT "First name: ", First$
      LOCATE 18, 40: PRINT "(Press Enter if n/a)"
      LOCATE 18, 25: INPUT "Area code: ", Area$
      LOCATE 20, 22: INPUT "Phone number: ", Phone$
      LOCATE 22, 22: INPUT "Company name: ", Compa$
      WRITE #1, Last$, First$, Area$, Phone$, Compa$: CLS
    END IF
  WEND
  CLOSE #1: Last$ = "": GOTO MainMenu
```

Listing 1 continued→

→ *Listing 1 continued*

```
CASE "2"        'Search for a name in the database
  WHILE UCASE$(Search$) <> "END"
    OPEN File$ FOR INPUT AS #1
    CLS : LOCATE 11, 25: PRINT "SEARCH for a name in the database"
      LOCATE 12, 25: PRINT "(To stop, enter END as the name)"
      LOCATE 14, 25: INPUT "Name to search for: ", Search$: PRINT
      IF UCASE$(Search$) <> "END" THEN
        DO WHILE (NOT EOF(1))
          INPUT #1, Last$, First$, Area$, Phone$, Compa$
          SELECT CASE UCASE$(Search$)
            CASE UCASE$(Last$)
              Hit% = 1: LOCATE , 25
              PRINT USING Form1$; Last$; First$;
              LOCATE , 25: PRINT USING Form2$; Area$; Phone$
              LOCATE , 25: PRINT USING Form3$; Compa$
            CASE UCASE$(First$)
              Hit% = 1: LOCATE , 25
              PRINT USING Form1$; Last$; First$;
              LOCATE , 25: PRINT USING Form2$; Area$; Phone$
              LOCATE , 25: PRINT USING form3$; compa$
            CASE UCASE$(Compa$)
              Hit% = 1: LOCATE , 25
              PRINT USING Form1$; Last$; First$;
              LOCATE , 25: PRINT USING Form2$; Area$; Phone$
              LOCATE , 25: PRINT USING form3$; Compa$
          END SELECT
        LOOP
        IF Hit% = 0 THEN
            LOCATE , 25: PRINT "No match found for "; Search$
        END IF
        Hit% = 0: LOCATE , 25: PRINT
        LOCATE , 25: PRINT "Press any key to continue"
        DO WHILE INKEY$ = "": LOOP
        CLOSE #1
      END IF
    WEND
    CLOSE #1: Search$ = "": GOTO MainMenu

  CASE "3"      'Quit the program
    CLS : SYSTEM

END SELECT


SUB Menu (Row, Column, Option$(), last, Selection) STATIC


  LMax = 0                                'Max length of options
  NumOpts = UBOUND(Option$)               'Number of options
  IF Selection = 0 OR Selection > NumOpts THEN Selection = 1

  FOR A = 1 TO NumOpts                    'Determine longest item
    B = LEN(Option$(A))
    IF B > LMax THEN LMax = B
  NEXT
```

Listing 1 continued →

search the file again is to open and close it for each search.

A second SELECT CASE block within the first locates a name in a file. The SELECT CASE block waits for the user to type the first name, last name, or company name (which is why you entered each element separately). When a match is found, the integer variable Hit% is set to 1, and all files matching the criteria are printed to the screen.

At the beginning of the program, the Form$ variables appear on three lines to help show how NAMES.BAS stores and retrieves each of the three search variables: Last$ = last name; First$ = first name; Compa$ = company name.

When no match exists, the value of Hit% is zero, and the program states that no match was found. A DO loop and INKEY$ again wait for user input. END completes the subprogram and returns you to the main menu.

Finally, the last section of SELECT CASE executes when the user presses 3. Then the program clears the screen and uses the SYSTEM statement to end the program.

Shortcomings

My NAMES.BAS database is not perfect. It changes the background screen after the first END command returns to the main menu, which some users might find awkward. In addition, unlike its commercial counterparts, NAMES.BAS does not let you sort names. Nor does it allow information to be changed or deleted. (It is an ASCII text file, so you can edit it in DOS 5.0's Edit screen editor. Bring up the file for editing simply by typing the following command:

```
C:\DOS\EDIT NAMES.TXT
```

(Assuming Edit is stored in your \DOS directory on drive C.) Just remember not to change any commas or quotation marks while editing.

Up and Running

Now that you've typed a QBasic program, why not write a batch file

→ Listing 1 continued

```
  LMax = LMax + 2                  'Add two for surrounding spaces

  FOR A = 1 TO NumOpts             'Display the choices
    COLOR FG, BG
    F = Row + A - 1
    LOCATE F, Column
    IF Option$(A) = "-" THEN
      PRINT STRING$(LMax, 196)
    ELSE
      PRINT " "; Option$(A); TAB(Column + LMax);

      B = INSTR(UCASE$(Option$(A)), A$)
      IF B THEN
        LOCATE F, Column + B
        PRINT MID$(Option$(A), B, 1)
      END IF
    END IF
  NEXT

  COLOR FG, BG
  LOCATE Row + Selection - 1, Column
  PRINT " "; Option$(Selection); TAB(Column + LMax);
    B = INSTR(UCASE$(Option$(Selection)), A$)
    IF B THEN
      LOCATE Row + Selection - 1, Column + B
      PRINT MID$(Option$(Selection), B, 1)
    END IF

  SELECT CASE T
       CASE -71                            'Home
         Selection = 1
       CASE -79                            'End
         Selection = NumOpts
       CASE -72
         Selection = Selection - 1         'Up arrow
         IF Selection = 0 THEN Selection = NumOpts
         IF MID$(Option$(Selection), 2, 1) = "-" THEN
           Selection = Selection - 1
         END IF
       CASE -80                            'Down arrow
         Selection = Selection + 1
         IF Selection > NumOpts THEN Selection = 1
         IF Option$(Selection) = "-" THEN
           Selection = Selection + 1
         END IF
       CASE ELSE
         Selection = Letter
     END SELECT

  END SUB
```

End

to run it? At the DOS prompt, type:

```
C:\DOS\EDIT NAMES.BAT.
```

When the DOS editor appears, type:

```
CD\DOS
QBASIC /RUN NAMES.BAS
CD\
```

This assumes that you will save the batch file NAMES.BAT in your root directory. NAMES.BAT changes to the DOS directory, starts QBasic, and runs NAMES.BAS. After you exit NAMES.BAS, the batch file brings you back to the DOS prompt at the root directory.

The simplest ways to expand NAMES.BAS involve altering the SELECT CASE block. You can create new fields and provide space for them in a Form$ variable. (But remember: Be certain to back up the existing program before making changes.)

Don't Keep It to Yourself

The batch file INSTALL.BAT (Program Listing 2) installs your QBasic program, copying it from a floppy disk to a hard disk. It places the BAS file in the \DOS directory and the batch file in the root directory. INSTALL.BAT is a little long, but it is easy to use and can be changed so that it will install almost any QBasic program you write. (The batch file listing includes instructions for making the appropriate modifications.) Simply copy the BAS program, the BAT file you use to start it, and INSTALL.BAT onto a single disk.

Then, you can easily pass disks along to friends to share the programs you create. If you share the customizing tips, too, others can take advantage of the database's flexibility to create useful variations on the basic theme. ■

David Bannon is a network administrator and computer programmer residing in South Carolina.

Program Listing 2. INSTALL.BAT lets you install the NAMES.BAS database and its accompanying batch file on your hard drive.

```
ECHO OFF
CLS
ECHO . HARD DRIVE INSTALLATION PROGRAM
ECHO . ---------------------------------------
ECHO . This program installs the program on your
ECHO   hard drive automatically.
ECHO .
ECHO . IMPORTANT:  Three things must be specified
ECHO   with the INSTALL command:
ECHO . (1)The PROGRAM on this disk to install on
ECHO   the hard drive
ECHO . (2)The HARD DISK DRIVE on which MS-DOS 5.0
ECHO   is located
ECHO . (3)THE PATH TO DOS on that drive.
ECHO . The proper syntax is:
ECHO .
ECHO .         INSTALL X Y Z
ECHO .         ... substitute program name for "X."
ECHO .         ... substitute letter of destination
ECHO           hard drive for "Y."
ECHO .         ... substitute path to DOS for "Z."
ECHO .
ECHO . Examples:   INSTALL NAMES C \DOS
ECHO . Your Entry: INSTALL %1 %2 %3
ECHO . ---------------------------------------
ECHO . Press Ctrl-C to cancel, or any other key
ECHO   to continue ...
PAUSE
CLS

IF EXIST \%1\*.* GOTO CHEKBAT
ECHO .

ECHO .         ERROR ... INSTALLATION ABORTED!
ECHO .
ECHO Install disk is not in the default drive,
ECHO or program was improperly specified.
ECHO Make the drive containing the
ECHO install disk the default drive and restart
ECHO this program.
GOTO END

:CHEKBAT
IF EXIST %2:\*.* GOTO CHEKDOS
ECHO .

ECHO .         ERROR ... INSTALLATION ABORTED!
ECHO .
ECHO The HARD DRIVE was incorrectly specified.
ECHO You may have misidentified
ECHO or failed to identify the destination
ECHO drive for this installation.
ECHO Check installation instructions.
ECHO .                                        ===
ECHO .                 Drive specified was ... %2
ECHO .                                        ===
GOTO END

:CHEKDOS
IF EXIST %2:%3\QBASIC.* GOTO CONTINUE
ECHO .

ECHO .         ERROR ... INSTALLATION ABORTED!
ECHO .
ECHO -  MS-DOS was not detected in directory %2:%3.
ECHO -
ECHO -  Drive and path to MS-DOS may have been
ECHO -  improperly specified. Check instructions.
ECHO -  -------------------------------------
ECHO .  IMPORTANT: Three things must be specified
ECHO    with the INSTALL command.
ECHO .  (1)The PROGRAM on this disk to install
ECHO    on the hard drive
ECHO .  (2)The HARD DISK DRIVE on which MS-DOS
ECHO    5.0 is located
ECHO .  (3)PATH TO DOS on that drive.
ECHO .  The proper syntax is:
ECHO .
ECHO .         INSTALL X Y Z
ECHO .         ... substitute program name for "X."
ECHO .         ... substitute letter of
ECHO           destination hard drive for "Y."
ECHO .         ... substitute path to DOS for "Z."
ECHO
ECHO . Examples:   INSTALL NAMES C \DOS
ECHO . Your Entry: INSTALL %1 %2 %3
ECHO . ---------------------------------------
GOTO END

:CONTINUE
ECHO STANDBY:  Installation in progress . . .
ECHO .
COPY \%1\*.BAS %2:%3
COPY \%1\*.BAT %2:\
CLS
ECHO This program was successfully installed
ECHO on drive %2.
GOTO END

:END
ECHO ON
```

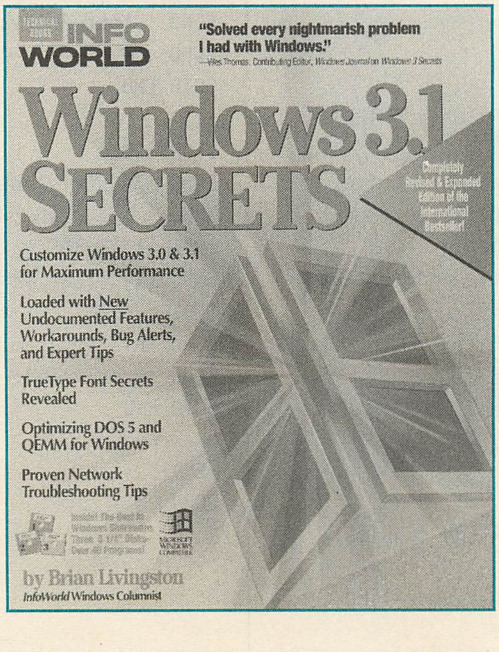
End

Using DOS Under Windows

Setting up your computer environment to allow Windows and DOS to coexist is only part of the battle. Getting them to actually work together is the other part.

by Brian Livingston

It's no secret: Windows sometimes has trouble coping with DOS sessions. But although understandable difficulties can crop up when Windows is forced to deal with ill-behaved DOS programs, you shouldn't give up hope. You can eliminate many potential problems simply by making sure DOS sessions are configured well for your computer system. You'll get not only smoother computing sessions but support for features DOS applications can't handle on their own (see the sidebar, "Getting the Windows Clipboard Back Under DOS").


INFO WORLD
"Solved every nightmarish problem I had with Windows."
Windows 3.1 SECRETS
Customize Windows 3.0 & 3.1 for Maximum Performance
Loaded with New Undocumented Features, Workarounds, Bug Alerts, and Expert Tips
TrueType Font Secrets Revealed
Optimizing DOS 5 and QEMM for Windows
Proven Network Troubleshooting Tips
by Brian Livingston
InfoWorld Windows Columnist

DOS Prompt Problems

Ironically, one of the programs that you must make sure not to run in a DOS session under Windows is WIN.COM itself. DOS's stark C:> prompt displayed against a bare expanse of black screen makes this an easy mistake to commit. After working at the DOS prompt for a while, it's perfectly natural to want to start Windows again by typing *WIN*. When Windows already is running, this can result in anything from an error message to hanging your system and forcing a reboot.

Windows 3.1 attempts to solve this problem in a variety of ways. When you start a DOS session under Windows 3.1, it displays a series of messages at the top of the first DOS screen. These messages warn you that you are starting a DOS session under Windows. They also tell you to type the EXIT command to end the DOS session or to press Alt-Tab to switch back to Windows without closing the DOS session.

This does not prevent you, however, from doing other things in a DOS session that you shouldn't do while Windows is running. For example, you might absent-mindedly start a DOS-based file-undelete or disk-opti-

***Editor's Note:** This discussion of Windows and DOS is excerpted from Brian Livingston's book* Windows 3.1 Secrets, *published by IDG Books Worldwide Inc., San Mateo, Calif., 1992. Reprinted with permission; all rights reserved.*

mization program. Neither Windows 3.1 nor 3.0 will protect you in cases like that.

You really need something that remains on the screen to remind you that you're operating in a DOS session under Windows. One way to keep your reminder on screen is to use the features of the DOS prompt. A new DOS prompt is written every time a DOS command is completed—after each DIR listing, for example.

You can use an undocumented feature of Windows 3.1 to alter your DOS prompt so it looks different when Windows is running. You probably know that you can put the command `SET PROMPT $P$G` in your AUTOEXEC.BAT so DOS always displays the current path and a greater-than sign after every command (as in the prompt `C:\WINDOWS>`, for example.)

With Windows 3.1, you can follow this statement in your AUTOEXEC.BAT with a second command:

```
SET WINPMT={prompt}
```

Windows 3.1 looks for this SET statement and uses any prompt that it finds there for its DOS sessions. When you exit Windows, DOS reverts to using the original SET PROMPT statement. You might, therefore, create an AUTOEXEC.BAT that looks something like the following program:

```
@ECHO OFF
PATH C:\;C:\WINDOWS;C:\DOS
SET PROMPT=$P$G
SET WINPMT=Type EXIT to quit DOS. - $P$G
WIN
```

When this program runs, it sets your prompts and starts Windows, ensuring DOS sessions you launch from Windows will use the WINPMT statement.

One possible reason this method wasn't documented by Microsoft is that it's implemented in a clumsy way. Windows 3.1 simply switches the two prompts with each other. If you type the command SET by itself in a DOS session, you will see the following lines:

```
PATH C:\;C:\WINDOWS;C:\DOS
PROMPT Type EXIT to quit DOS. - $P$G
WINPMT $P$G
```

This shows that the prompts were reversed. But it's not entirely accurate, because the WINPMT statement you set is not "PG," it's "Type EXIT to quit DOS — PG." People who have learned to use the SET command by itself can become confused when they see this output on the screen, because it's accurate only when Windows is in control. When you exit Windows, the prompts return to normal.

One very good thing about the SET WINPMT statement is that you can get rid of the lines that Windows 3.1 writes at the top of the DOS screen every time you start a DOS session (which gets annoying after about the second time). You can do this using another command that isn't in your Windows manual. Place the following in the [386Enh] section of your SYSTEM.INI file:

```
DOSPromptExitInstruc=False
```

This disables the lines that Windows writes, giving you a clean screen to work on when you start DOS sessions. (The lines don't appear in standard mode, since you can't have more than one DOS session in standard mode and can't start WIN.COM from a DOS session in standard mode, either.)

The Universal Fix

Unfortunately, neither the SET WINPMT trick nor the DOSPromptExitInstruc command work in Windows 3.0. You might like to develop a method to keep the status of DOS and Windows in view at all times, in whatever version of Windows is running. If that is the case, you should understand some of the basic features of running DOS sessions under Windows.

Perhaps you've considered that a good way to remind yourself (under any version of Windows) that you're running a DOS session under Windows is by starting a batch file from the Program Manager that changes your DOS prompt from a simple C:> to:

```
Type EXIT to quit DOS. - C>
```

You then try to start a batch file, such as:

```
ECHO OFF
PROMPT=Type EXIT to quit DOS. - $P$G
```

In this batch file, PROMPT should change what you see in the DOS session to include your exit message, followed by the full pathname of the current directory ($P) and a greater-than sign ($G). But your PROMPT command fails, and you see an "Out of environment space" error message. And then your DOS session closes by itself, throwing you back to Windows.

What happened? There's nothing in the Windows manual about this. Can't you run a batch file from Windows? Sure you can. To explain this mystery and a work-around for it, I'll first describe a fix for the environment space, then how to make a DOS session that was started from a batch file "stick."

Preserving the Environment

The DOS environment is a memory area used to store the PATH and PROMPT statements in your AUTOEXEC.BAT files. You also can put any other kind of information there, then read this area of memory later using batch files. Whatever is presently stored in the environment is displayed when you type the SET command at the DOS prompt. If you issue the command:

```
SET LABEL=1234
```

then type *SET* by itself, you see that `LABEL=1234` is part of the environment, along with your PATH and PROMPT statements. These text strings that you set in the environment are called environment variables.

DOS sets the maximum size for this environment area when it runs your CONFIG.SYS file. Unless you specify otherwise, this memory area is limited to 160 bytes (256 bytes in DOS 5.0). This usually is not enough, so I recommend that you always specify at least 512 bytes by making the first line of your CONFIG.SYS say the following:

```
SHELL=C:\COMMAND.COM /E:512 /P
```

The specification C:\COMMAND.COM must state the exact location of your copy of COMMAND.COM. The /E:*nnn* parameter sets the environment size in bytes. (If you use DOS 3.0 or 3.1, the number after /E: represents blocks of 16 bytes, so to get 512 bytes of environment space, you specify /E:32, which is 512 divided by 16.) The parameter /P is required and makes COMMAND.COM load itself permanently, instead of exiting immediately.

If your environment previously was limited to only 160 bytes, however, this was not the cause of the "Out of environment space" error message you got when trying to set a different prompt in a DOS session. The environment is even more limited than this under Windows, unless you know how to correct it.

When Windows (or any DOS application) starts, it receives from DOS a copy of whatever is in the environment at that moment. To give the application as much memory as possible, DOS doesn't hand it the entire 160-byte environment space (or whatever you set the value to). It only gives the application that part of the environment that has some text values in it, plus a few extra bytes that round it up to an exact multiple of 16 bytes. If 60 characters currently are in the environment, DOS adds four blank bytes to give each application 64 bytes, a multiple of 16.

When you start Windows, it receives this part of the environment, as any DOS application would. And when you start a DOS session under Windows, your DOS session has only that same environment size to work with—64 bytes, or whatever the present limit is. Therefore, it can never have more than 15 bytes free in which to set environmental variables, and usually it has fewer.

You could fix the DOS prompt under Windows by starting Windows from a batch file (call it W.BAT) and,

GETTING THE WINDOWS CLIPBOARD BACK UNDER DOS

The Windows Clipboard is an extremely useful area of memory for Windows applications. Unfortunately, it doesn't work perfectly with DOS sessions.

In Windows applications, highlighting text and pressing Ctrl-Ins copies the text into the Clipboard. Pressing Shift-Del has the effect of deleting the text from the application while moving it to the Clipboard. In either case, moving the cursor to another location or another application, then pressing Shift-Ins, pastes the text into the new location. After performing any of these actions, you can view the contents of the Clipboard memory area by running the CLIPBRD.EXE program included with Windows. This program actually is a Clipboard viewer, not the Clipboard itself. The Clipboard memory area can contain many types of data other than text—bit-mapped graphics, Windows metafile graphics, and so on.

DOS applications, however, vary widely in their support for the Windows Clipboard. Many DOS programs are totally oblivious to the existence of the Clipboard (or, for that matter, of Windows). But other programs, such as Microsoft Word for DOS Versions 4.x and 5.x, have choices right on their menus for copying to and pasting from the Clipboard. (This assumes that Windows is running and, therefore, that the Clipboard exists.)

These menu choices fail if Windows is running in real mode or standard mode. Windows 2.x (either the 286 or the 386 version) always made pasting from the Clipboard available to DOS applications; but Windows 3.0 "lost" this ability, except if you're working in 386 Enhanced mode. If you know the secret, though, you can get the Clipboard back—even in old DOS applications that are totally unaware such a thing as the Clipboard exists.

Under all three modes in Windows 3.x, you can copy text from a DOS application into the Clipboard simply by pressing the PrintScreen key. (You lose the ability to use PrintScreen to send the DOS application's screen to the printer under Windows 3.0 in real and standard modes.)

But to paste text from the Clipboard into a DOS application under real or standard modes, you must use the following work-around:

before Windows loads, setting the prompt that you want to display in your DOS sessions. The following batch file would make any DOS session display the "Type EXIT to quit DOS." message discussed earlier:

```
ECHO OFF
PROMPT=Type EXIT to quit DOS. - $P$G
WIN
PROMPT $P$G
```

The first PROMPT statement sets the prompt used under Windows. The second one (which runs after you have exited Windows) sets the prompt back to the normal style you want to use in DOS when Windows isn't running. There are several more elegant ways to set the prompt in a DOS session, which I discuss later on.

Fixing the prompt still doesn't relieve a different type of environment-space problem in a DOS session. Say you want to run a batch file from Windows that sets an environmental variable then starts an application that reads that variable to configure itself. In Windows 3.0, your batch file's SET LABEL command would fail, just as the prompt did in the first example.

Windows 3.1 handles this situation correctly, actually reading the SHELL statement you placed in your CONFIG.SYS file to see how much extra environment space (if any) you established. When you start a DOS session under Windows 3.1, it gives a DOS session an environment space as large as the one you originally had under DOS. (You must use DOS 3.2 or later for this to work.)

If this amount of environment space isn't enough, you can use another obscure SYSTEM.INI command to force Windows to give your DOS sessions even more. Place the following line in the [NonWindowsApp] section of your SYSTEM.INI file:

```
CommandEnvSize=1024
```

This command allocates 1,024 bytes of conventional memory to the environment space of each DOS session you start. (You can choose any value you want, but it probably should be a multiple of 16 bytes, as in the SHELL command in CONFIG.SYS.)

By using a line like this is your SYSTEM.INI, you could use a smaller environment number in your SHELL statement in CONFIG.SYS (thereby saving a few bytes of memory for ordinary DOS applications, if you're that tight on memory). But you could have all the environment space you want in DOS sessions under Windows.

The CommandEnvSize statement does not work in Windows 3.0, of course. The answer to the environment space problem that will work for all versions of Windows—my personal environmental preservation act—is to create an environment space in a DOS session that is large enough to support the addition of environmental variables. You do this by running COMMAND.COM from Windows with a switch like one used in the SHELL command described earlier.

Specifically, to start a DOS session with an environment size of 512 bytes, pull down Program Manager's File Run menu, then type:

```
COMMAND /E:512
```

Step 1. Place the cursor in your DOS application at the point where you want text from the Clipboard to be inserted.

Step 2. Switch to the Windows application that contains the text you want to transfer, using Alt-Tab-Tab to see a list of your currently running applications. (Hold down the Alt key, press the Tab key repeatedly, then release the Alt key when the name of the desired application appears. If your BIOS does not support Alt-Tab, use Alt-Esc.)

Step 3. In the Windows application, highlight the text you want to transfer, and press Ctrl-Ins to copy it to the Clipboard.

Step 4. Find the DOS icon representing your DOS session. This icon is now minimized on the icon line at the bottom of your Windows Desktop, since you previously switched away from your DOS session.

Step 5. Click once on the DOS icon to pop up its Control menu. Click the Paste choice on the menu. (If you haven't copied anything to the Clipboard first, Paste won't appear as a valid choice on this menu, which is why most people don't know it's there.)

Step 6. Double-click the DOS icon to restore it to its previous full-screen, foreground status. *Voila!* Your selected text has been inserted into your DOS application, exactly as if you had typed it yourself. If your highlighted text ends with a carriage return, you even can feed complicated DOS commands to a C:> prompt in this way.

If you are running Windows in 386 Enhanced mode, Paste will appear on the menu of DOS icons as Edit Paste. Additionally, in 386 mode, your DOS application might be running in a window instead of occupying a full screen. In that case, you don't have to minimize the DOS application to an icon to use this method. Instead, click once on the control bar in the upper-left corner of the sizable window. Then choose Edit Paste on the drop-down menu that appears.

Also in 386 mode: If the text that appeared in your DOS application is missing a few characters, the program may not be able to receive keystrokes as fast as the Clipboard is capable of sending them. In this instance, you need to run this application from a PIF file in which you have turned off the Allow Fast Paste option.

—B.L.

In this line, there is no /P parameter. You are not starting an original, permanent instance of COMMAND.COM. You want this copy to go away when you type the EXIT statement to quit the DOS session and return to Windows. Hence, it is temporary. (If you are using DOS 3.0 or 3.1, the number after /E: is not the number of bytes, but the number of 16-byte chunks as described earlier, and you should change this number accordingly.)

When you run this line from Windows' File Run dialog box, a DOS session starts, and you get a black screen with whatever prompt was in effect before Windows loaded. But now, you have an environment space in which to work. Type:

```
SET JUNK=ABCDEFGHIJKLMNOP
```

(or any 16 letters), then type SET by itself. All the letters you typed are now stored in the environment. Then, simply type:

```
SET JUNK=
```

with nothing after the equal sign, and type the SET command as before. You have eliminated the variable Junk from the environment, and the space it formerly used is available again for other variables.

Instead of typing this into the File Run dialog box every time you want a DOS session, you should put this command line into a Program Information File (PIF) and run that instead. By defining a DOS session icon that runs this PIF, you will create an environment space that always will be large enough to meet your computing needs.

Using the PIF Editor to create this PIF file (call it DOS.PIF) is fairly simple. You need to know only the following settings:

```
Command Line: C:\COMMAND.COM
Window Title: DOS Session
Optional Parameters: /E:512
```

The Survival of Batch Files Under Windows

Now that you understand the procedures for setting prompts and starting a DOS session with a large enough environment space, you can start a batch file that configures that environment. As you recall, when you originally tried to run a batch file under Windows to set the prompt and then leave you at that DOS prompt to work, your batch file was terminated immediately and you were thrown back to Windows.

This happened because after the last line of your batch file was executed, Windows assumed (correctly) that batch processing was finished and reclaimed the memory used by that session.

The secret to making this batch file survive, giving you that Zen-black screen in which to work away at a DOS prompt, is to make sure the last line of the batch file contains the equivalent of the COMMAND instruction. The best way to accomplish this is to use an environmental variable that you may already have seen when you ran the SET command: the so-called COMSPEC variable.

When you turn on your PC, DOS itself sets an environmental variable, named COMSPEC, equal to the path specification in which COMMAND.COM may be found. This usually looks like:

```
COMSPEC=C:\COMMAND.COM
```

Many things can change the location of the COMMAND.COM file that you execute—especially networks. But no matter where COMMAND.COM is located for you, the COMSPEC variable identifies that location.

Therefore, to make a batch file survive, make the last line equal to the environmental variable COMSPEC, as shown:

```
ECHO OFF
PROMPT Type EXIT to quit DOS. - $P$G
%COMSPEC%
```

When this batch file runs, DOS interprets any item between percent signs (%) as an environmental variable and inserts the value of that variable. If your COMSPEC is equal to C:\COMMAND.COM, this batch file is translated by DOS into:

```
ECHO OFF
PROMPT Type EXIT to quit DOS. - $P$G
C:\COMMAND.COM
```

Using %COMSPEC% at the end of a batch file is a safer method to use than hard-coding the word *COMMAND* into the batch file, since the environment variable works in all cases and in all PC configurations. Notice that there is no need to use parameters after *%COMSPEC%* or *COMMAND*—the word itself is all you need.

Since this batch file adds to the DOS environment (because it makes the prompt longer), you must start it from a copy of COMMAND.COM that enlarges the environment, just as I discussed earlier when I explained how to run COMMAND.COM from the File Run dialog box. To run this batch file (or any such batch file), add its name to the end of the parameters when you run COMMAND.COM.

If this batch file is named MYPROMPT.BAT and is located in a directory named in your PATH statement, the command line you would use in File Run would look like this:

```
COMMAND /E:512 /C MYPROMPT
```

In a PIF file, the same command would be specified with the following lines:

```
Command Line: C:\COMMAND.COM
Window Title: DOS Session
Optional Parameters: /E:512 /C MYPROMPT
```

You can place any DOS program (those with COM, EXE, or BAT extensions) after the /C in this command line. That program will load with its own enlarged environment space. If this environment space isn't needed, simply run the DOS program directly from its own PIF file, without first starting a secondary copy of COMMAND.COM.

Detecting That Windows Is Running

Once your DOS session is running, you may have several different batch files that your system offers. However, you want to make sure that none of these batch files runs dangerous DOS applications (such as CHKDSK.COM with its /F parameter) if the batch file happens to be in a DOS session at the time. How can a batch file detect this?

If you ran the SET command in a DOS session during one of the experiments discussed earlier in this article, you may have noticed a curious string stored in the environment—one you know you didn't place there. This variable probably looked like this (the capitalization is exactly as you would see it on your computer screen):

```
windir=C:\WINDOWS
```

When Windows is running, it places this variable in your DOS environment. The variable windir is set to the directory that contains WIN.COM. This usually is the directory that contains all your Windows files. But if you, like many people, have moved WIN.COM and all INI and GRP files to another directory to keep writable data files out of the Windows program directory (and then placed that directory on your PATH statement in front of the main Windows directory), the windir variable will be set equal to that directory.

With this knowledge, you might think that a batch file could test whether or not it was running in a DOS session under Windows by checking for the existence of a variable named windir, as this short program does:

```
ECHO OFF
IF NOT "%windir%"=="" GOTO :ERROR
REM Perform dangerous
REM DOS tasks here.
GOTO :END
:ERROR
ECHO You cannot use this batch file while Windows
ECHO is running.
:END
```

In this batch file, the IF NOT statement tests whether the variable %windir% exists. The percent signs identify windir as an environment variable. The quotation marks ensure that the batch file has something on both sides of the double equal signs, in case %windir% is blank.

This batch file, however, will fail to detect the windir environment variable, even if Windows is running—because both Windows 3.0 and Windows 3.1 set this variable using lowercase letters. In DOS, environment variables always are converted to uppercase, so your batch file tests for the string *WINDIR* not *windir*.

If you are an experienced DOS user and conversant in Debug, you can force Windows to place this variable in the environment in capital letters. But you needn't go to all the trouble, because there's an easy and safe

Program Listing 1. ISWIN.SCR is a Debug script you use to create the executable program ISWIN.COM, which provides a reliable method of testing whether you are in a DOS session under Windows. ISWIN.COM and ISSHARE.COM (see Program Listing 2) were developed by Fran Finnegan of Finnegan O'Malley & Company Inc., a Windows consulting and contract programming firm and the developer of E-Mail Manager. These programs were adapted with permission from Windows Q&A, by Fran Finnegan, *Microsoft Systems Journal*, March 1991. Copyright Microsoft Corp.

```
N ISWIN.COM
A 100
MOV     AX,4680
INT     2F
XOR     AL,80
MOV     CL,AL
MOV     AX,1600
INT     2F
AND     AL,7F
OR      AL,CL
CMP     AL,80
JZ      011A
MOV     AH,4C
INT     21
MOV     AX,1605
XOR     BX,BX
MOV     ES,BX
XOR     SI,SI
MOV     DS,SI
XOR     CX,CX
MOV     DX,0001
MOV     DI,0300
INT     2F
CMP     CX,+00
JNZ     0139
MOV     AX,1606
INT     2F
MOV     AL,80
OR      AL,CL
MOV     AH,4C
INT     21

RCX
41
W
Q
```

End

ISWIN.COM'S ERROR LEVELS

Errorlevel Value	Meaning
0	Windows is not running
1	Windows/386 2.x is running
3	Windows 3.x is running 386 enhanced mode
4	Windows 4.x is running 386 enhanced mode
127	Windows/386 2.x is running
128	Windows 3.x is running real mode
255	Windows 3.x is running standard mode

way to test for Windows: running a program called ISWIN.COM.

Using ISWIN.COM

You can create this program by typing Program Listing 1 into a text editor. Leave a blank line after the INT 21 instruction and press Enter after the Q command in the last line. Save the file in a directory mentioned in your DOS PATH statement, naming it ISWIN.DEB.

Then, issue a `DEBUG<ISWIN.DEB` command at the DOS prompt. Debug reads your file and creates the program ISWIN.COM, which provides a reliable method for telling whether you are in a DOS session under Windows. It even can tell you which mode and version of Windows is running.

When you run ISWIN.COM, it sets a variable known as the DOS Errorlevel. DOS programs and other applications normally set this variable, which may be equal to any value from zero to 255, inclusive. Programs usually set the Errorlevel equal to zero if their routines were completed normally, and equal to 1 or other numbers if errors were encountered.

ISWIN.COM sets the Errorlevel according to values shown in the table "ISWIN.COM's Error Levels." Since all these Errorlevel values are set to 1 or more if any version of Windows is running, you could jump out of a batch file, if necessary, by running ISWIN.COM and then testing the result with a single command, as in:

```
ECHO OFF
ISWIN
IF ERRORLEVEL 1 GOTO :ERROR
REM Place your "dangerous" DOS command here.
GOTO :END
:ERROR
ECHO You can't use this batch file
ECHO while Windows is running.
:END
```

In this batch file, the statement `IF ERRORLEVEL 1` means "if the Errorlevel is 1 or more, carry out the rest of this line."

Testing the Errorlevel value in a batch file doesn't clear the value. It remains valid until another DOS program sets it again. So, if you need to know which mode or version of Windows is running, you can test the Errorlevel several times in a batch file and branch to different commands. The command:

```
IF ERRORLEVEL 3 IF NOT ERRORLEVEL 4 DOTHIS
```

for example, runs the program DOTHIS only if the Errorlevel is equal to 3 (indicating Windows 3.x in enhanced mode).

While you're testing for Windows, you also should test whether SHARE.EXE was loaded before Windows. As you may know, SHARE.EXE must be loaded in your AUTOEXEC.BAT file to prevent the corruption of your files when two or more applications are running.

You can test for the presence of SHARE.EXE by using an even simpler program than ISWIN.COM. Use a text editor to type in the Debug script shown in Program Listing 2—it's only 11 lines long. Save the file in a directory mentioned in your DOS PATH statement, calling it ISSHARE.DEB.

Then, type `DEBUG<ISSHARE.DEB` at the DOS prompt. Running the resulting ISSHARE.COM program sets the DOS Errorlevel to 1 (yes) if SHARE.EXE is loaded and to zero (no) if it is not.

Now you can use ISSHARE.COM in the W.BAT batch file that starts Windows. The following W.BAT, for example, checks to see whether SHARE.EXE is loaded before starting Windows. If it isn't, the IF statement loads it:

```
ECHO OFF
ISSHARE
IF NOT ERRORLEVEL 1 SHARE /F:2048 /L:20
WIN
```

Brian Livingston, president of Windows Consulting, leads seminars on Windows at the Windows and OS/2 Conference sponsored by CM Ventures.

Program Listing 2. ISSHARE.SCR is a Debug script you use to create the executable file ISSHARE.COM, which tests for SHARE.EXE.

```
N ISSHARE.COM
A 100
MOV    AX,1000
INT    2F
MOV    AH,4C
INT    21

RCX
9
W
Q
```

End

Q I have read that APPEND and FASTOPEN cause problems for Windows. Everyone skips the explanation. Do you have one? Also, if these commands are bad for Windows and, if DOS and Windows are both Microsoft products, why would the company allow such a situation to continue?

A APPEND and FASTOPEN, like certain other memory-resident programs, interfere with the installation or running of Windows. APPEND interferes with the ability of Windows and its applications to build valid paths for files they want to access. FASTOPEN, on the other hand, works in most instances, but the information it supplies is redundant if an efficient cache is used. It also takes up memory that Windows otherwise could use effectively.

Popular cache programs include SMARTDRV.SYS, which is bundled with Windows 3.0 and DOS 5.0; its improved replacement, SMARTDRV.EXE, which is included in Windows 3.1; and the excellent Super PC-Kwik cache (PC-Kwik Corp., 800-274-5945, $79.95).

JOIN and SUBST are two other DOS commands that require special care under Windows. You can invoke these commands prior to starting the software, but you must not change them from within a DOS window. Commands that alter the logical relationship among drives, such as ASSIGN, should not be used at all.

Unfortunately, eliminating incompatibilities and inconsistencies is not possible without introducing new and different problems. Each version of DOS maintains backward compatibility with those that came before. If these commands were altered or eliminated, then DOS-based programs relying on their actions would fail.

When you install Windows, Setup warns about possible problem conditions and urges their correction. Unfortunately, Microsoft could not anticipate every condition or program that may affect Windows, so conflicts can occur. This shouldn't be a problem, however, after the company completes its development of a protected-mode DOS operating system in one to two years.

Q I have just upgraded my computer from a 16MHz 286 to a 16MHz 386SX, using an Everex board. I have 4MB of RAM and another 6MB configured as expanded memory, and I am running DOS 4.0. Everything is slower, not faster. Why is that? Also, I can't seem to get into the 386 enhanced mode. Will I have to reinstall Windows?

A The Everex upgrade to which you refer is a small board that plugs into the CPU socket on the motherboard. This upgrade can handle a maximum of 4MB of motherboard memory; additional memory must be provided on memory cards that plug into your system's expansion slots.

You may be using an expanded-memory manager that the manufacturer of your memory card provided. Windows will not start in 386 enhanced mode when some memory managers are active. Also, Windows cannot utilize expanded memory. In this case, the additional 6MB should be configured as extended memory.

Further, DOS 4.0, although adequate for your previous computer setup, is a poor relation to the newer DOS 5.0, which offers memory management through HIMEM.SYS and EMM386.EXE. If you have programs that need expanded memory, EMM386 can provide it without preventing Windows from running in its enhanced mode. You also can use QEMM (Quarterdeck Office Systems, 800-354-3222, $99.95), 386MAX (Qualitas, 301-907-6700, $99.95), or one of several other memory management programs.

by Jan Norman Johnsen

Power user and computer consultant Jan Norman Johnsen has been involved with personal computers since 1978. He was a beta tester for Windows 3.0 and has done compatibility testing for Microsoft Corp. on MS-DOS 5.0 and Windows 3.1.

Q Can you suggest a way that I might create "hot keys" or macros in Windows to get from one application (or group of applications) to another? Can I do this with DOS applications running under Windows?

A With Windows 3.1, you can accomplish this easily from each application's Properties dialog box. (To call up this box, select the program's icon with a single click, click on the File option in the menu bar, then select the Properties option.) You'll see a space in which to type the key you want to assign to call up a particular application. (The exact hot-key sequence will be Ctrl-Alt plus the key you choose.) The Program Information (PIF) editor provides a similar box in its advanced section.

Another way to toggle between open programs is to use Alt-Tab. This method, which works under Windows 3.0 as well as Windows 3.1, switches you between programs that are open (underneath the application you are working on) or to applications that are represented by icons at the bottom of the desktop.

A third way is to task switch using the Task Manager, which is activated by double-clicking on an empty part of the desktop or by pressing Ctrl-Esc. As with Alt-Tab, this technique works with Windows 3.0 and Windows 3.1.

Q Can you provide a formula for figuring out the optimal amount of memory Windows needs? If I dedicate more memory to it, will that increase the speed of redrawing a window?

A To operate Windows efficiently, I recommend a minimum of 4MB of memory, even though the software can limp along on about 1.5MB in enhanced mode, if you start it with the following command:

```
WIN /3
```

I consider 8MB (the additional 4MB should cost less than $200) of memory to be desirable, however. The additional memory will not increase the speed at which program screens are redrawn, although it should significantly speed up swapping when multiple applications are in use.

To view your system's memory resources, I recommend purchasing Memory Viewer in First Apps (hDC Corp., 206-885-5550, $99.95) or the Systems Resources Monitor (Microsoft, 800-642-7676), which is part of the Windows 3.1 Resource Kit ($19.95).

Q Is it wise to load my mouse driver into high memory? I'm trying, but so far I've had no luck. How should I do this?

A You can load your mouse driver into high memory, provided it is designed to be loaded high. When the driver is loaded high, it leaves more memory for programs and the individual applications you run.

But, as a rule, drivers written prior to the release of DOS 5.0 may not be designed for loading high. Check the date of the driver or contact the manufacturer or your dealer to find out whether yours may be loaded high. Note that Setup (under Windows 3.0 or 3.1) may have updated the driver automatically during installation.

To load the Microsoft mouse driver high in DOS 5.0, use the following CONFIG.SYS line, changing the path if necessary:

```
DEVICEHIGH=C:\WINDOWS\MOUSE.SYS /Y
```

If you load your mouse driver in AUTOEXEC.BAT, the line for loading a mouse high should read:

```
LH C:\WINDOWS\MOUSE.COM /Y
```

The /Y parameter is needed for some video configurations; I recommend including it to prevent possible problems.

Either of these commands should ensure that your mouse driver is loaded into the upper memory block (UMB), if sufficient space is available. Although some recent versions of Microsoft's mouse driver may be loaded into the high memory area (HMA), I don't recommend this because, at present, Microsoft doesn't support this.

Q I recently ran the Norton Disk Doctor program and discovered some bad clusters on my hard drive. So I moved each to a new location and optimized the drive. I began, thereafter, to have some problems with my Windows applications and had to reinstall them. I'm still having problems, however, and I think I will reinstall Windows. Do you have any advice? What steps should I take to save my setups?

A The behavior you mention resembles that caused by some viruses, so I recommend you check your hard disk carefully with a good virus-detection program. Secondly, it's possible that your disk's original formatting is weak or somehow damaged. Since you have Disk Doctor, which is a part of The Norton Utilities package (Symantec Corp., 408-253-9600, $179), I suggest you run the company's Calibrate program or use SpinRite II (Gibson Research, 714-362-8800, $89) to correct problems with the low-level formatting. Then optimize the disk.

If you have upgraded to Windows 3.1, some of the old Windows 3.0 setup configurations may be causing the problem. In that case, reinstalling Windows in a new directory should eliminate the problem.

But if the difficulty recurs, I would suspect an impending hard disk failure. All hard disks eventually fail, and timely replacement may be the best strategy. ■

Now available to readers of *DOS Resource* from...

DevWare Direct

IN OUR SEVENTH YEAR!

and..

TECHMEDIA DISCOUNT

FALL BLOWOUT!

ENTERTAINMENT SOFTWARE ADVENTURE/SIMULATION

Aces of the Pacific (Sierra On-Line)	E1001	$49
A-10 Tank Killer 1.5 (Sierra On Line)	E1002	$41
Airborne Ranger (MicroProse)	E1003	$15
Bane of the Cosmic Forge (Sir-Tech)	E1004	$39
Bane of Cosmic Forge II:Crusaders	E1005	$47
Battle Isle (EA)	E1006	$37
B-17 Flying Fortress (MicroProse)	E1007	$41
Birds of Prey (EA)	E1008	$37
Bravo Romeo Delta	E1009	$39
Carrier Strike (SSI)	E1010	$47
Castles II:Siege and Conquest (InterPlay)	E1011	$43
Challenge of the 5 Realms (MicroProse)	E1012	$47
Chuck Yeager's Air Combat (EA)	E1013	$41
Covert Action (MicroProse)	E1014	$29
Crisis in the Kremlin	E1015	$45
Darklands (MicroProse)	E1016	$43
Dark Queen of Krynn (SSI)	E1017	$39
Defender of the Crown (Cinemaware)	E1018	$12
Dragon Lord (Cinemaware)	E1019	$12
Dune (Virgin Games)	E1020	$37
Elvira II:Jaws of Cerberus (Accolade)	E1021	$41
Eye of the Beholder II (SSI)	E1022	$41
F-15 Strike Eagle (MicroProse)	E1023	$12
F-117A (MicroProse)	E1024	$49
Faery Tale Adventure (Hollyware)	E1025	$19
Faery Tale Clue Book (Hollyware)	E1026	$6
Free D.C. (CinePlay)	E1028	$37
Great Naval Battles:North Atlantic (SSI)	E1030	$39
Gunboat (Accolade)	E1031	$24
Gunship 2000 (MicroProse)	E1032	$45
Gunship (MicroProse)	E1033	$12
Hyperspeed (MicroProse)	E1034	$41
James Bond:Stealth Affair (InterPlay)	E1035	$22
Kings Quest V (Sierra On-Line)	E1036	$49
Knights of the Sky (MicroProse)	E1037	$29
Lightspeed (Microprose)	E1252	$39
Lost Treasures of Infocom (Activision)	E1038	$47
Magic Candle 2 (Mindcraft Software)	E1039	$41
Martian Memorandum (Access)	E1040	$41
MegaTraveller 2 (MicroProse)	E1041	$39
Midwinter (MicroProse)	E1042	$19
MiG-29 Fulcrum (Domark)	E1043	$34
Might and Magic 3 (New World Computing)	E1044	$39
Might and Magic 4 (New World Computing)	E1045	$43
No Greater Glory (SSI Civil War)	E1046	$43
Obitus (Psygnosis)	E1047	$29
Pacific Islands	E1048	$34
Patton Strikes Back (Broderbund)	E1049	$39
Planet's Edge (New World Computing)	E1050	$41
Realms (Virgin Games)	E1052	$33
Rings of Medusa (Digitek)	E1053	$27
Savage (MicroProse)	E1054	$12
SDI (Cinemaware)	E1055	$12
Second Conflict	E1056	$37
Second Front: Germany Turns East (SSI)	E1057	$39
Shuttle (Virgin Games)	E1060	$37
Siege (Electronic Arts)	E1061	$41
Silent Service II (MicroProse)	E1062	$37
Silent Service I (MicroProse)	E1063	$12
Solo Flight (MicroProse)	E1064	$12
Star Glider II (MicroProse)	E1065	$12
Space 1889 (MicroProse)	E1066	$12
Strike Aces (Accolade)	E1067	$19
Team Yankee II (ReadySoft)	E1068	$33
The Battle of Britain (Lucasfilm)	E1069	$39
Thunderstrike (Live Studios)	E1070	$21
Ultima Trilogy II (Origin)	E1072	$49
Vengeance of Excaliber (Virgin Games)	E1073	$25
Wizardry Trilogy (Sir-Tech Software)	E1074	$33
Wing Commander II (Origin)	E1075	$49

Power Politics for Windows from Cineplay Interactive is a complete simulation of the presidential election, allowing up to three players to pit their presidential candidates against each other and find ou;who wins the '92 election, or play agains the computer. A1096 $29

Free D.C! is the first computer game to use Claymation characters combined with human actors and Cinetalk (Cineplay's system to synch recorded speech), in a true multimedia experience. E1028 $37

EDUTAINMENT

Annabel's Dream - Ancient Egypt (Texan Caviar)	D1141	$57
Designasaurus II (Britannica)	D1142	$29
Hare Raising Havoc (Disney Software)	D1143	$33
Headline Harry and Paper Race (Davidson)	D1144	$33
Katie's Farm (Broderbund)	D1145	$17
KidPix (Broderbund)	D1146	$33
KidWorks (Davidson & Associates)	D1147	$35
Math Maze (Britannica)	D1148	$29
Mixed Up Fairy Tales (Sierra On-Line)	D1150	$33
My Paint (Saddleback Graphics)	D1151	$35
My Paint - Alphabet Fun Coloring Disk	D1152	$15
My Paint - Majelix Coloring Disk	D1153	$15
PC Globe V.5.0 (PC Globe)	D1157	$45
Storybook Weaver (MECC)	D1159	$35
Where in Europe - Carmen (Broderbund)	D1161	$35
Where in America's Past - Carmen (Broderbund)	D1162	$39
Zoo Keeper (Davidson & Associates)	D1166	$41

ENTERTAINMENT SOFTWARE ACTION/ARCADE/TRADITIONAL

Air Ball (MicroDeal)	A1076	$12
Atomino (Psygnosis)	A1078	$29
Battle Chess II (InterPlay)	A1079	$33
Battle Chess (InterPlay)	A1080	$29
Big Business (DigiTek)	A1081	$27
Corporation (Virgin Games)	A1084	$41
Dino Wars (DigiTek)	A1085	$24
Dr. Doom's Revenge (MicroProse)	A1086	$12
Future Classics Collection (Live Studios)	A1087	$19
Global Effect (Millennium)	A1088	$37
Heaven and Earth (Buena Vista Software)	A1089	$34
Leisure Suit Larry 5 (Sierra On-Line)	A1090	$47
Lemmings (Psygnosis)	A1091	$31
Nevermind (Psygnosis)	A1092	$15
Oh No! More Lemmings (Psygnosis)	A1093	$34
Populous (EA)	A1094	$19
Power Monger (EA)	A1095	$34
Power Politics for Window (CinePlay)	A1096	$29
Railroad Tycoon (MicroProse)	A1097	$29
Romantic Encounters (Hollyware)	A1099	$12
Sim Earth (Maxis)	A1102	$45
Solitaire's Journey (QQP)	A1103	$41
Space Ace I (ReadySoft)	A1104	$33
Space Ace II:Borf's Revenge (ReadySoft)	A1105	$37
Speedball 2 (Konami)	A1106	$33
Star Control (Accolade)	A1107	$34
Super Tetris (Spectrum Holobyte)	A1109	$37
Troika (MicroProse)	A1111	$18
Wild Wheels (EA)	A1112	$34
Wordtris (Spectrum Holobyte)	A1113	$29

ENTERTAINMENT SOFTWARE SPORTS

Al Michaels:Hard Ball III (Accolade)	A1114	$39
4-D Sports Boxing (EA)	A1115	$33
Bill Elliott's NASCAR Challenge (Konami)	A1116	$33
Carl Lewis Challenge (Psygnosis)	A1117	$39
Earl Weaver Baseball II (EA)	A1118	$37
Grand Prix Unlimited (Accolade)	A1119	$39
HardBall II (Accolade)	A1120	$32
Heatwave:Superboat Racing (Accolade)	A1121	$22
Hole-in-One Miniature Golf (DigiTek)	A1122	$24
International Sports Challenge (ReadySoft)	A1123	$31
Jack Nicklaus' Unlimited Golf (Accolade)	A1125	$33
John Madden Football (EA)	A1126	$27
Jordan In Flight (EA)	A1127	$47
Links (Access)	A1128	$39
Mike Ditka Ultimate Football (Accolade)	A1129	$37
Mario Andretti's Racing Challenge (EA)	A1130	$32
MicroLeague Football:Deluxe Edition	A1131	$47
NFL Challenge - Premium Edition (XOR)	A1132	$59
Oh No! More Lemmings (Psygnosis)	A1133	$33
Pro Football Analyst (Villa Crespo)	A1134	$39
Pro Tennis Tour 2 (EA)	A1135	$37
Road to Final Four (Bethesda Softworks)	A1136	$41
Stunts (Broderbund)	A1137	$32
Test Drive III (Accolade)	A1138	$41
Tony Larussa's Ultimate Baseball (SSI)	A1139	$34
3-D Pool (MicroProse)	A1140	$12

PRODUCTIVITY

Business Contacts (Disk-Count)	P1167	$49
Data Manager (Timeworks)	P1168	$29
DeskTop Set 3.0 (Okna Corp.)	P1169	$105
Eight in One (Spinnaker)	P1170	$42
First Publisher (Spinnaker)	P1171	$105
GeoWorks Desktop (GeoWorks)	P1172	$49
GeoWorks Pro (GeoWorks)	P1173	$139
Instant Recall 2.0 (Chronologic Corp.)	P1174	$89
Microsoft Works (Microsoft)	P1175	$105
Newsroom Pro (Spinnaker)	P1176	$29
OnTime for Windows 3.1 (Campbell Services)	P1177	$89
Personal R:Base 1.0 (Microrim)	P1178	$71
PFS:Write (Spinnaker)	P1179	$89
ProCalc 3-D (Parsons Technology)	P1180	$65
Publish It (Timeworks)	P1181	$99
Publish It Lite (Timeworks)	P1182	$42
Resident Expert - Effective Writing (Parsons)	P1183	$41
Quicken (Intuit)	P1184	$54
SideKick 2.0 (Borland Int'l)	P1185	$69
WordWriter Pro (Timeworks)	P1186	$69

UTILITIES

Andrew Tobias (MECA)	U1187	$63
Becker Tools 3 for Windows (Abacus)	U1254	$95
Brooklyn Bridge (Fifth Generation Syst.)	U1188	$99

Order Toll-Free! (800) 879-0759

Turn the page to see the rest of our products and order form.

Carbon Copy Plus (Microcom)	U1189	$139
Central Point Backup 7.1 (Central Point)	U1190	$89
Disk Optimizer (SoftLogic Solutions)	U1191	$49
Devware Utility Disk - 15 programs	U1192	$20
Double Density (Abacus)	U1253	$59
Fastback Plus 3.04 (Fifth Generation Syst.)	U1193	$129
Laplink Pro (Traveling Software)	U1194	$119
Netroom 2.0 (Helix)	U1195	$67
NoMouse for Windows (Abacus)	U1255	$39
Norton Backup (Symantec)	U1196	$89
Norton Utilities 6.01 (Symantec)	U1197	$129
OPTune 1.3 (Gazelle Systems)	U1198	$69
PC Kwik Power Disk 1.1 (Multisoft Corp.)	U1199	$55
PC Tools 7.1 (Central Point)	U1200	$125
QEMM-386 6.02 (Quarterdeck Office Syst.)	U1201	$69
QMAPS 2.1 (Quantel Corp.)	U1202	$69
Resident Expert - All DOS Versions (Parsons)	U1203	$33
Spinrite (Gibson Research)	U1204	$63
Tempest (Abacus)	U1205	$25
386MAX 6.01 (Qualitas)	U1206	$69
ViruCide Plus (Parsons Technology)	U1207	$49
Viruscure (IMSI)	U1208	$69

Graphics

Animation Studio (Disney Software)	G1209	$89
DeluxePaint Animation (Electronic Arts)	G1210	$95
New Print Shop (Broderbund)	G1211	$43
PC Paintbrush IV (Z Soft)	G1212	$139
Will Vinton's Playmation for Windows (CinePlay)	G1213	$299

Programming

Dr.DOS v6 (Digital Research)	R1214	$69
MS-DOS 5.0 (MicroSoft)	R1214	$69
Windows 3.1 (MicroSoft)	R1215	$109
FinancialDome PLUS (Dome)	R1216	$65
Money Counts 7.0 (Parsons Technology)	R1217	$37
Quicken/Turbo Tax (Intuit)	R1218	$109
Peachtree Basic Accounting (Peachtree)	R1219	$79
Personal Portfolio Manager (Abacus)	R1220	$69

Miscellaneous

Animation Video Vol. I	M1221	$15
Animation Video Vol. II	M1222	$19
Basic Programming Inside Out - Book & Disk	M1223	$27
Batch File Power Tools - Book with Disk	M1224	$29
Computer Viruses & Data Protection - Book	M1225	$15
DeluxePaint Animation Video	M1226	$24
DOS 5.0 Complete - Book with Disk	M1227	$33
DOS 5.0 Essentials - Book	M1228	$15
Family Origins (Parsons)	M1229	$35
It's Legal (Parsons)	M1230	$51
Laser Printer Power Tools - Book	M1231	$24
Mind's Eye Animation Video	M1232	$19
PC Assembly Language - Book with disks	M1233	$29
PC Intern - Book	M1234	$44
Personal Advocate (Parsons)	M1235	$51
The 486 Book - with Disk	M1236	$29
Tips and Tricks for Your PC Printer - Book	M1237	$24
VideoMaker Video	M1238	$25
HardwareCable for modems	M1239	$11
Cable for printers	Call	$11
Joysticks		$Call
Sound Blaster (Creative Labs)	M1241	$119
Sound Blaster Pro (Creative Labs)	M1242	$219
Sound Blaster Speakers	M1243	$24
SupraModem 2400 - external	M1244	$95
SupraModem 2400 Plus - external	M1245	$139
SupraFaxModem 2400/9600 Plus	M1246	$164
SupraFaxModem V32 bis	M1247	$379
SupraModem 2400 internal	M1248	$75
SupraModem 2400 Plus internal	M1249	$105
Thunder Board (Media Vision)	M1250	$95

Books

Basic Programming: Inside & Out (Abacus) w/disk	B1270	$27
Batch File Powertools (Abacus) w/disk	B1271	$27
DOS 5.0 for Beginners (Abacus)	B1272	$15
Dr. DOS 6.0 (Abacus) w/disk	B1273	$29
The OS/2 Book (Abacus) New!	B1274	$19
PC Assembly Language: Step by Step (Abacus) 2 disks	B1275	$27
QBasic for Beginners (Abacus) New!	B1276	$16
The Sound Blaster Book (Abacus) w/disk -- New!	B1277	$31
Turbo C++: Step by Step (Abacus) w/disk -- New!	B1278	$27
Turbo Pascal for Windows (Abacus) w/disk	B1279	$31
Turbo Vision Toolbox (Abacus) w/disk	B1280	$31
Wicked Windows (Abacus) New!	B1281	$17
Windows 3.1 Complete (Abacus) w/disk	B1282	$27
Windows 3.1 Intern (Abacus) w/disk -- New!	B1283	$31
WordPerfect for Beginners (Abacus)	B1284	$15

Hot Manufacturer of the Month: *Microprose*

F-117A - All the features of the award-winning F-19 Stealth Fighter, plus more missions and worlds, spectacular night graphics with special HUD features and improved Super 3-D graphics and improved artifical intelligence make this the most realistic and challenging simulator yet! E1024 $49

Gunship 2000 - Take total command of a full multi-copter troop of America's most powerful rotor craft. You take charge of five copters on hundreds of combat missions in the Persian Gulf and Central Europe... E1032 $45

B-17 Flying Fortress - The most complete and accurate bomber simulation ever produced, B-17 Flying Fortress will have you negotiating 25 perilous daylight missions over Nazi-occupied Europe. E1007 $41

Hot Title!

Will Vinton Studios, the Oscar-winning creators of Claymation and "The California Rasins", has announced a new breakthrough in 3D computer animation. **Will Vinton's Playmation for Windows** is the first 3D software with no polygons, incorporating a revolutionary new process that brings Hollywood-style computer special effects to the PC for both novices and pro's alike. P1213 $299

AutoMap - The intelligent road atlas shows you the right route to drive by searching through more than 350,000 miles of road and over 50,000 U. S. cities in seconds with full color maps and detailed directions you can print out and take along. P1256 $67

Special: *DevWare Productivity Disk*

Best Seller--Super Value! Make every computing operation smoother and make yourself more productive when you employ Cruiser and the 14 other utility programs on this special disk. Cruiser allows you to see inside any file on your disk and page through it as if the disk were one large book. Cruiser has a text editor, pop-up calculator, and much more! U1192 $20

Top 10 Games

1.	Ultima VII (Origin)	E1071	$49
2.	Secret Weapons of the Luftwaffe (Lucasfilm)	E1058	$47
3.	SimCity (Maxis)	A1101	$34
4.	Star Trek 25th Anniversary (Interplay)	A1108	$39
5.	Civilization (Microprose)	A1083	$44
6.	Risk (Virgin Games)	A1098	$34
7.	Police Quest III (Sierra On-Line)	E1051	$43
8.	SimAnt (Maxis)	A1100	$39
9.	Teenage Mutant Ninja Turtles - Arcade (Konami)	A1110	$31
10.	American Gladiators (Gametek)	A1077	$27

Top 10 Home Education

1.	Where/World is Carmen Sandiego (Broderbund)	D1165	$33
2.	Where/U.S.A. is Carmen Sandiego (Broderbund)	D1164	$34
3.	Mavis Beacon Teaches Typing (Software Toolworks)	D1149	$39
4.	New Math Blaster Plus (Davidson)	D1154	$35
5.	Playroom (Broderbund)	D1156	$37
6.	Where/Americas Past/Carmen Sandiego (Broderbund)	D1162	$39
7.	Oregon Trail (MECC)	D1155	$35
8.	Treehouse (Broderbund)	D1160	$41
9.	Where in Time is Carmen Sandiego (Broderbund)	D1163	$35
10.	Reader Rabbit 2 (Learning Company)	D1158	$41

Return policy: All programs in origional packaging. Refunds only on product price. Call for Return Authorization Number.

ORDER BY TELEPHONE OR MAIL IN THIS FORM

ORDER FORM

Mail this order form to:
DEVWARE DIRECT,
12520 Kirkham Court, Suite 1-DR1
Poway, CA 92064

Shipping Charges
US: 3% of total order ($5 minimum)
Canada: 4% of total order ($6 minimum)
Foreign: 6% of total order ($10 minimum)
C.O.D. add $5 (U.S. orders only)
All payments in U.S. funds only.
Prices subject to change without notice.
30-Day money back guarantee on all products.

All games require 512K and DOS 2.2+.
Please specify disk size (5.25" or 3.5").

Name ______________________________
Address ______________________________
City ______________ State ______________ Zip ______________
Country ______________________________
Day Phone #( ) ______________________________

Qty.	Product ID/Name	Unit Price	Total

7.75% Sales Tax (CA Res. only) ________
Shipping ________
Total ________

Method of Payment
☐Check/Money Order
☐Visa/MasterCard
☐Discover

Credit Card # ______________________________
Expiration ____/____ Signature: ______________________________

Order Toll-Free! (800) 879-0759

Customer Service: (619) 679-2825
Fax your order to: (619) 679-2887

SUBSCRIBE TO DOS RESOURCE GUIDE 800-238-6580

AND SAVE 20% OFF

COVERS ALL DOS VERSIONS
DOS RESOURCE GUIDE
The PC Productivity Magazine
PRESENTS:
THE ESSENTIAL GUIDE TO DOS
2nd Edition
NOW INCLUDES DOS 6.0
•Batch File Tips and Tricks
•MANAGING MEMORY STEP-BY-STEP
•COMPLETE GUIDE To DOS Files
DOS
DOS RESOURCE GUIDE
The PC Productivity Magazine
THE COMPLETE DOS Q&A:
Files and Directories
DOS Setup
Improved Productivity
Memory Management
Hardware Upgrades
GET STARTED WITH QBASIC
DOS RESOURCE GUIDE
The PC Productivity Magazine
THE ONE-STOP DOS PRIMER
All About DOSKEY
APPEND, ASSIGN, And SETVER Can End Compatibility Problems
Fine-tune Your CONFIG.SYS
The 10 Worst DOS Commands
DEBUG Almost Defined
INSIDE: 15 DOS TIPS
PROTECT YOUR DATA FROM VIRUS ATTACK

THE COVER PRICE

BUCK THE TREND! STAY WITH DOS

Subscribe NOW and get 6 issues of DOS RESOURCE GUIDE for $23.70.* Save a buck an issue off the newsstand price!

If you're having trouble understanding DOS, stop worrying. It's not you!

Understanding DOS is the key to effective PC-computing. But if you've tried to learn DOS on your own, wrestled with its manual, purchased a guidebook, or taken a course, you're probably still frustrated and confused.

Relax, you're not alone!

One estimate puts the number of DOS-run computers in the marketplace at 70 million. But how many DOS experts are there?

Darn few, and here's why.

Since late 1980, Microsoft Corp.'s DOS, the essential software ingredient of the IBM PC-compatible , has changed many times. Along the way, the operating system that began as 22 cryptic commands has grown to about 100 equally cryptic commands, drivers, and programs.

DOS RESOURCE GUIDE explores DOS's hidden secrets, untangles the twisted technical prose of manuals, and tells you in clear language how to use DOS.

In each issue of DOS RESOURCE GUIDE, you'll find articles about using DOS commands, about managing your computer's memory, or maintaining order on your hard disk.

Beginners will find step-by-step help at copying files, creating and removing directories, using DOS's wildcards to cut down on keystrokes, and practical advice on using DOS's many "switches" after each command.

More advanced users will learn about batch files, which can provide capabilities that DOS forgot, or automate tedious activities, such as moving or backing up large groups of files. And you'll learn to use DOS 5's new QBasic programming language, and create macros at DOS's command line.

Each issue of DOS RESOURCE GUIDE is filled with tips on using DOS, undocumented shortcuts, and batch files you can type in and use.

So, tap all of your computing potential by learning DOS, personal computing's most important—if not most popular— operating system. Order today! Use the coupon below, or call:

800-238-6580

YES! I'M BUCKING THE TREND! YES! I'M STAYING WITH DOS!

☐**YES!** **Send me 6 Issues of DOS Resource Guide for $23.70.***

NAME ____________________

ADDRESS ____________________

CITY ____________________ STATE ________ ZIP ________

(Check, Money Order, or Credit Card—Payment must accompany your order.)

☐ AMEX ☐ VISA ☐ MASTERCARD ☐ DISCOVER

CREDIT CARD # ____________________

EXPIRATION DATE: ____________________

SIGNATURE: ____________________

MAIL TO: DOS Resource Guide, 80 Elm Street, Peterborough, NH 03458

*In Canada and Mexico pay $29.70; all other foreign (Surface) $41.70 (Air) $53.70.

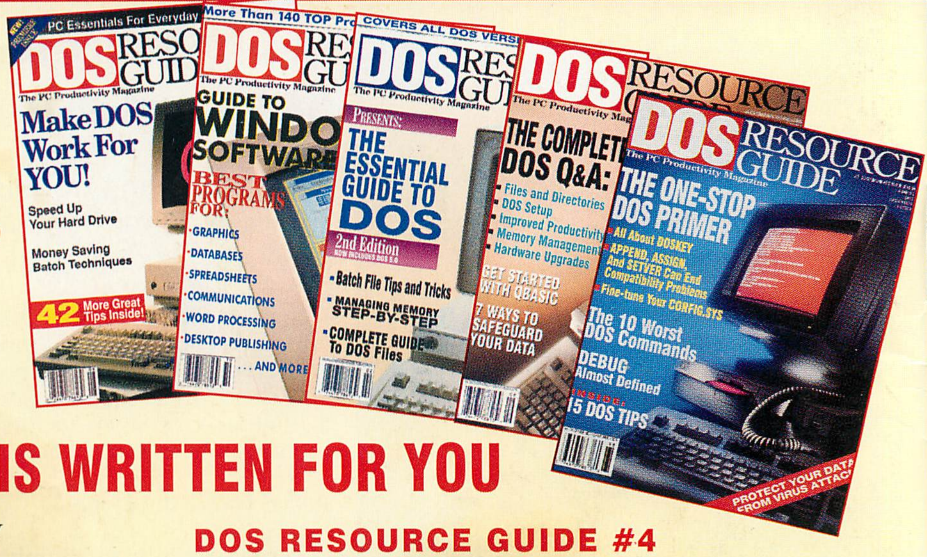
Checks and money orders must be payable in U.S. funds, drawn on a U.S. bank.

Maryland residents add 5% sales tax.

HV6AD

TAKE THE DREAD OUT OF DOS!

If you're trying to succeed with DOS, but feel like a General Failure...



DOS RESOURCE GUIDE IS WRITTEN FOR YOU

DOS RESOURCE GUIDE, the bi-monthly periodical packed with tips, tutorials, and information on all aspects of PC computing, is your antidote for computing confusion.

DOS RESOURCE GUIDE #1

Make DOS Work for You

Covers six areas of computing that are vital to DOS users: start-up files (AUTOEXEC.BAT and CONFIG.SYS), batch file creation, hard drives, safe backups, organizing file management, and hardware maintenance. Includes: List'em, a batch file that helps you keep appointments; 10 tips to keep your hard drive healthy; speed-up tips for a logy hard drive; a batch file that automates backups; DOS shortcuts, hardware dos and don'ts. Plus 42 provocative pointers on DOS.

DOS RESOURCE GUIDE #2

Guide to Windows Software

SOLD OUT

A compendium of software briefs that tells you what's up in the world of Windows. (Even a DOS user has to keep up with the competition.) Guide #2 briefly describes over 140 programs for Microsoft Windows, including word processors, communications programs, financial and math/stat programs, management programs, developer's tools, and graphics software. Complete list of manufacturers.

DOS RESOURCE GUIDE #3

The Essential Guide to DOS

Who are we to argue with success! Our second edition of this classic is updated to include information on DOS 5. Guide #3 includes an annotated list of DOS files and when to use them; articles that will change your view of CHKDSK, COPY, and XCOPY; a cluster of batch file assists; and some clear-headed advice on organizing files on your hard drive. Plus 14 titillating tips on DOS.

DOS RESOURCE GUIDE #4

The Complete DOS Q&A

Breaks DOS into functional sections: Fundamentals, Set-up, Improving Productivity, Memory Management, Hardware Upgrades, Batch Files, and Using Windows. Each section is followed by a question and answer session on the topic. Plus, seven short programs to protect your data; do-it-yourself memory, advice on CONFIG.SYS and AUTOEXEC.BAT, and beginner's Basic. Not to mention nine tips for the DOS timid.

DOS RESOURCE GUIDE #5

The One-Stop DOS Primer

All you can eat: DOSKEY, SETVER, APPEND and ASSIGN, plus a tutorial on using the ATTRIB command. Part 1 of a Debug series. Our "10 Worst" list of DOS commands. Your introduction to serial ports. Includes a Qbasic Printer Control Program: Just type and run. Sharpen your DOS finesse with 15 type-in tips.

Back issues of DOS Resource Guide are $5.95* postpaid!

Use our 800 number and have your credit card ready!

1-800-238-6580

Or send your check or money order to:

DOS Resource Guide
P.O. Box 3162
Salisbury, MD 21802

*Maryland residents add 5% sales tax. Back issues to Canada and Mexico cost $6.95 (Postpaid). Other International (Surface) $8.00 (Air) $13.00. Checks and money orders must be payable in U.S. funds, drawn on a U.S. bank.

VISA, Mastercard, or American Express also accepted. Please include your card number, its expiration date, and your signature on your request.

Please indicate which issue(s) of DOS Resource Guide you are ordering.

DON'T MISS AN ISSUE!